PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-139140
(43)Date of publication of application : 20.05.1994

(51)Int.Cl. G06F 12/06
G06F 12/00

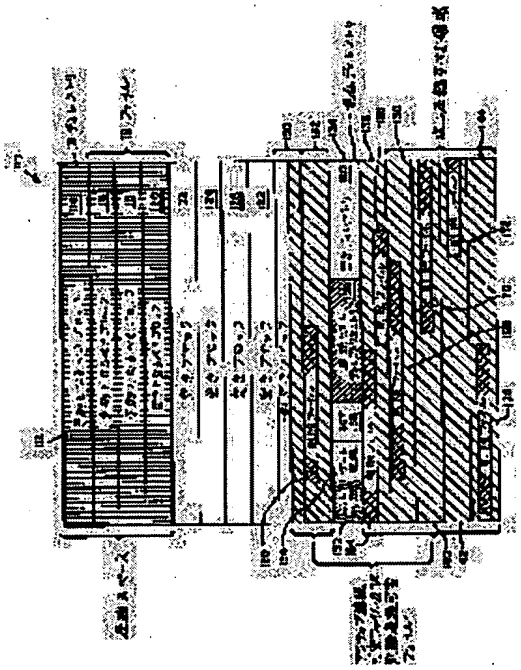
(21)Application number : 03-356593
(22)Date of filing : 25.12.1991

(71)Applicant : INTEL CORP
(72)Inventor : ROBINSON KURT B
ELBERT DALE K
LEVY MARKUS A

(30)Priority
Priority number : 90 636238 Priority date : 31.12.1990 Priority country : US

(54) FILING STRUCTURE OF NONVOLATILE SEMICONDUCTOR MEMORY

(57)Abstract:
PURPOSE: To enable erasure of bit only block as unit and to overwrite a bit from 1st logical state to 2nd logical state without erasing any bit in advance by providing an active block, reserved block and directory block.
CONSTITUTION: While a personal computer system is operated, data or a code is introduced into an empty block, a block 130 is a partially empty block and the active code is contained at one part. Besides, blocks 114, 116 and 118 of a flash memory array 112 are reserved blocks and during a clean-up operation, temporary file backup is executed. Besides, a directory block 134 includes a directory flag 152, boot record 154 and block state table 156 or the like. Then, when the boot record 154 is to be made invalid, a logical value '1' in the total inspection data field of the boot record 154 is overwritten into logical value '0'.



LEGAL STATUS

[Date of request for examination] 26.11.1998

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平6－139140

(43)公開日　平成６年(1994)５月20日

(51)Int.Cl.[5]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	12/06	５１０	9366－5B		
	12/00	５２０　Ｊ	8526－5B		

審査請求　未請求　請求項の数５（全 35 頁）

(21)出願番号　特願平3－356593

(22)出願日　平成３年(1991)12月25日

(31)優先権主張番号　６３６２３８

(32)優先日　1990年12月31日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　591003943
インテル・コーポレーション
アメリカ合衆国　95052　カリフォルニア州・サンタクララ・ミッション　カレッジ　ブーレバード・2200

(72)発明者　カート・ブライアン・ロビンソン
アメリカ合衆国　95658　カリフォルニア州・ニューキャスル・ネバス　レイン・2216

(74)代理人　弁理士　山川　政樹

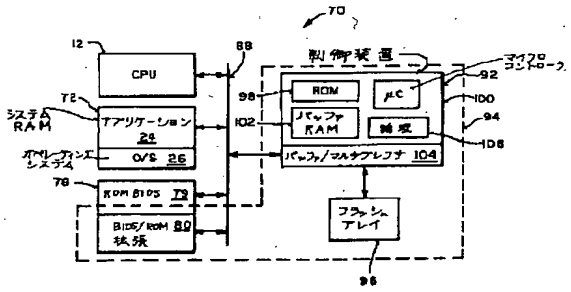
最終頁に続く

(54)【発明の名称】　不揮発性半導体メモリのファイル構造

(57)【要約】

【目的】　ブロック単位でのみ消去可能であるが、第１の論理状態から第２の論理状態に重ね書きすることは不可能である不揮発性半導体メモリのファイル構造の提供。

【構成】　不揮発性半導体メモリは第１のファイルを記憶するアクティブブロックと、第２のファイルを記憶する予約ブロックと、ディレクトリブロックとを具備する。第２のファイルは第１のファイルのコピーである。そのコピーはアクティブブロックの消去に先立ってクリーンアップ動作中に作成される。ディレクトリブロックは、第１のファイルを識別するためのディレクトリエントリから成る。



【特許請求の範囲】
【請求項１】　ブロック単位でのみ消去可能である不揮発性半導体メモリであって、その各ビットをあらかじめ消去せずに第１の論理状態から第２の論理状態に重ね書きすることは不可能であるが、各ビットをあらかじめ消去せずに第２の論理状態から第１の論理状態に重ね書きすることは可能であるような不揮発性半導体メモリにおいて、
（Ａ）第１のファイルを記憶するアクティブブロックと；
（Ｂ）アクティブブロックの消去に先立ってクリーンアップ動作中に作成される第１のファイルのコピーである第２のファイルを記憶する予約ブロックと；
（Ｃ）第１のファイルを識別するためのディレクトリエントリから成るディレクトリブロックとを具備する不揮発性半導体メモリ。
【請求項２】　（Ａ）ブロック単位で消去可能である不揮発性半導体メモリであって、その各ビットをあらかじめ消去せずに第１の論理状態から第２の論理状態に重ね書きすることは不可能であるが、各ビットをあらかじめ消去せずに第２の論理状態から第１の論理状態に重ね書きすることは可能であり、
（１）第１の複数のファイルを記憶するアクティブブロックと；
（２）アクティブブロックの消去に先立ってクリーンアップ動作中に作成される第１の複数のファイルのコピーである第２の複数のファイルを記憶する予約ブロックとを含む不揮発性半導体メモリと；
（Ｂ）それぞれが第１の複数のファイルの中の対応する１つのファイルを識別する複数のディレクトリエントリを記憶するメモリとを具備するコンピュータのメモリシステム。
【請求項３】　（Ａ）第１の複数のファイルを記憶するアクティブブロックと；
（Ｂ）アクティブブロックの消去に先立ってクリーンアップ動作中に作成される第１の複数のファイルのコピーである第２の複数のファイルを記憶する予約ブロックと；
（Ｃ）（１）ディレクトリブロックをディレクトリブロックとして識別するためのフラグと、（２）フラッシュＥＥＰＲＯＭのファイル構造に関する情報を含むブート記録と、（３）ブロックが（ａ）　アクティブブロックであるか又は予約ブロックであるか、（ｂ）　未使用であるか又は使用中であるか及び（ｃ）　欠陥を含んでいるか又は含んでいないかに関する情報を提供するブロック状態テーブルと、（４）それぞれが第１の複数ファイルの中の対応する１つのファイルを識別する複数の連係リストディレクトリエントリとを含むディレクトリブロックとを具備するフラッシュ電気的消去可能プログラマブル読取り専用メモリ（ＥＥＰＲＯＭ）。
【請求項４】　（Ａ）中央処理装置と；
（Ｂ）各ビットをあらかじめ消去せずに第１の論理状態から第２の論理状態に重ね書きすることは不可能であるが、各ビットをあらかじめ消去せずに第２の論理状態から第１の論理状態に重ね書きすることは可能であり、
（１）第１のファイルを記憶するアクティブブロックと、（２）アクティブブロックの消去に先立ってクリーンアップ動作中に作成される第１のファイルのコピーである第２のファイルを記憶する予約ブロックと、（３）第１のファイルを識別するためのディレクトリエントリから成るディレクトリブロックを含むブロック単位で消去可能な不揮発性半導体メモリと；
（Ｃ）不揮発性半導体メモリを制御するコードを記憶する記憶手段とを具備するコンピュータシステム。
【請求項５】（Ａ）中央処理装置と；
（Ｂ）各ビットをあらかじめ消去せずに第１の論理状態から第２の論理状態に重ね書きすることは不可能であるが、各ビットをあらかじめ消去せずに第２の論理状態から第１の論理状態に重ね書きすることは可能であり、
（１）第１の複数のファイルを記憶するアクティブブロックと、（２）アクティブブロックの消去に先立ってクリーンアップ動作中に作成される第１の複数のファイルのコピーである第２の複数のファイルを記憶する予約ブロックとを含むブロック単位で消去可能な不揮発性半導体メモリと；
（Ｃ）それぞれが第１の複数のファイルの中の対応する１つのファイルを識別する複数のディレクトリエントリを記憶するメモリと；
（Ｄ）不揮発性半導体メモリを制御するコードを記憶する記憶手段とを具備するコンピュータシステム。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】本発明はコンピュータ記憶システムのアーキテクチャの分野に関する。特に、本発明は大形ブロック消去可能不揮発性半導体メモリのファイル記憶システムに関する。
【０００２】
【従来の技術】従来のある種のパーソナルコンピュータシステムは、数種類の記憶システムである読取り専用メモリ（「ＲＯＭ」）、ランダムアクセスメモリ（「ＲＡＭ」）、大容量記憶のためのハード（すなわち、固定）ディスクドライブ及び出入れ自在磁気フロッピーディスクへの記憶のためのフロッピーディスクドライブに結合するマイクロプロセッサ（中央処理装置ともいう）を含む。フロッピーディスクはディスケットとも呼ばれている。このような従来のパーソナルコンピュータシステムは、通常、それぞれのパーソナルコンピュータシステムの一部を形成する記憶システムに特別に適応するアーキテクチャを有する。
【０００３】ＲＯＭと、ＲＯＭに記憶されているＲＯＭ

モニターと呼ばれるプログラムとを併せてファームウェアという。ＲＯＭベーシング入出力システム（ＢＩＯＳ）モジュールはＲＯＭに記憶されて、ある種のパーソナルコンピュータについてオペレーティングシステムにより使用されるＲＯＭモニターの１例である。ＲＯＭＢＩＯＳモジュールは、通常、（１）　キーボード、ディスクドライブ及びプリンタを含めたいくつかのハードウェアのドライバと；（２）　パワーオン自己試験プログラム（「ＰＯＳＴ」）と；（３）　システムを初期設定するスタートアップルーチンと；（４）　ディスケット又はハードディスクからブート、すなわち第１のセクターを読取るローダプログラムとを含む。

【０００４】パーソナルコンピュータがオンされた後、ＢＩＯＳモジュールのＰＯＳＴプログラムが実行され、ＢＩＯＳスタートアップルーチンはいくつかの初期設定を実行し、ローダプログラムはディスケット又はハードディスクからブートセクターの内容を読取る。そのブートセクターはコンピュータのオペレーティングシステムのローダプログラムを含む。オペレーティングシステムのローダプログラム自体は、ディスケット又はハードディスクからＲＡＭへオペレーティングシステムの一部をロードする。

【０００５】パーソナルコンピュータのオペレーティングシステムは指令を処理し、プログラムの実行を制御し、コンピュータシステムのハードウェア資源及びソフトウェア資源を監視する。従来のオペレーティングシステムの１つの型がワシントン州レドモンドのマイクロソフト・コーポレーションから販売されているＭＳ－ＤＯＳである。ＭＳ－ＤＯＳオペレーティングシステムは先に述べたローダプログラムと、ＭＳ－ＤＯＳ　ＢＩＯＳと、ＭＳ－ＤＯＳカーネルと、ユーザーインタフェースと、ユーティリティプログラムとを含む。

【０００６】ＭＳ－ＤＯＳ　ＢＩＯＳは（１）　ＲＯＭ　ＢＩＯＳのドライバの構成要素を拡張し且つ使用する新たなドライバと；（２）　ＭＳ－ＤＯＳ　ＢＩＯＳドライバの初期設定ルーチンと；（３）　別のローダプログラムとを含む。

【０００７】ＭＳ－ＤＯＳ　ＢＩＯＳの新たなドライバをＢＩＯＳ拡張又はＢＩＯＳ　ＲＯＭ拡張ともいう。従来のパーソナルコンピュータの中には、ＲＯＭの３２キロバイト部分又は６４キロバイト部分及びそれより小さい（通常は２キロバイトの）ＢＩＯＳ　ＲＯＭ拡張に記憶されているソフトウェアによって全ての入出力機能を駆動するものがある。

【０００８】ＭＳ－ＤＯＳ　ＢＩＯＳドライバの初期設定ルーチンはコピーライト公報メッセージを表示すると共に、新たなドライバに関する割込みテーブルを調整する。ＭＳ－ＤＯＳ　ＢＩＯＳのローダプログラムはオペレーティングシステムの残る部分をロードする。

【０００９】ＭＳ－ＤＯＳカーネルはＢＩＯＳとアプリケーションプログラムとの間のシェルである。ＭＳ－ＤＯＳカーネルはアプリケーションプログラムの実行を開始させ、アプリケーションプログラムのためのメモリを割付け、アプリケーションプログラムとハードウェアとの間にアプリケーションプログラムインタフェースを構成し且つファイルの読取りと書込みを管理する。

【００１０】ＭＳ－ＤＯＳのユーザーインタフェースはユーザーに情報を提供する。ユーザーインタフェースは、ユーザーに指令の入力を促すプロンプトを与える。オペレーティングシステムが制御中であるとき、ユーザーインタフェースはシステムのマネジャとして動作する。

【００１１】ＭＳ－ＤＯＳのユーティリティプログラムはＭＳ－ＤＯＳに関するいくつかの有用な機能を提供する。それらの機能には（１）　ディスケット又はハードディスクの定様式化及び（２）　ディスケット又はハードディスクの検査がある。

【００１２】先に述べたようなパーソナルコンピュータシステムと共に使用されるハードディスクやディスケットは不揮発性記憶システムである。すなわち、コンピュータへの給電が停止したときにデータは失われないのである。ハードディスク及びディスケットはブロック記憶装置の１種である。すなわち、データはブロック単位で転送される。

【００１３】ハードディスクやディスクの場合、データは同心のトラックに物理的に記憶される。それぞれのトラックは複数のセクターから構成されている。通常、セクターは５１２バイトの長さで固定されている。パーソナルコンピュータシステムのディスク制御装置と物理装置ドライバは、常にディスクのセクターエントリとの間でデータの書込み及び読取りを実行するのが普通である。

【００１４】クラスタは論理的にアドレッシング可能な最小の記憶単位である。ハードディスクによっては、クラスタごとに４つのセクターを含む。他のハードディスクや高密度３．５インチディスケットの場合には、１つのクラスタた単一のセクターである。

【００１５】ハードディスクの各区画領域は、その独自のオペレーティングシステムを記憶することができる論理サブシステムを形成する。定様式化ハードディスクの第１のセクターにある区画テーブルは、区画に関する情報を含む。

【００１６】ハードディスクとディスケットは、最初に使用される前に定様式化される。低レベルの定様式化は各トラックを複数のセクターに分割し、識別（「ＩＤ」）セクターヘッダをトラックに沿って均一に配分された位置に配置する。高レベル定様式化はクラスタを確定し、いくつかのディスク領域を初期設定し、データを受信するためにディスクの準備を整える。

【００１７】装置ドライバ及びＢＩＯＳレベルでは、デ

ィスク要求は、ドライブ，ヘッド，シリンダ／トラック，セクター及び長さを指示する「テュープル」により記述される。ＤＯＳレベル及びＢＩＯＳレベルでは、論理セクター番号はわかっている。ＤＯＳはディスクのクラスタエントリとの間で読取り及び書込みを実行する。

【００１８】図１は、従来のＭＳ－ＤＯＳオペレーティングシステムの論理的編成を示す。ＭＳ－ＤＯＳオペレーティングシステムの場合、ディスク２はシステム領域４と、データ領域９という２つの論理領域に分割されている。システム領域はブートレコード３と、ファイル割付けテーブル（「ＦＡＴ」）５と、ルートディレクトリエントリを含むルートディレクトリ７とを含む。データ領域９は、アプリケーションプログラム、データ及びサブディレクトリ情報を記憶するために使用されるファイルを含む。

【００１９】ブートレコード３は、オペレーティングシステムをロードするブートストラップローダプログラムを含む。ブートレコード３は、定様式化ＤＯＳのＡＳＣＩＩ名と、ディスクのセクターごとのバイト数と、クラスタごとのセクター数と、ブートレコード中のセクター数と、ファイル割付けテーブルのコピーの数と、ルートディレクトリエントリの数と、区画ごとのセクターの数と、ディスク型番号と、ファイル割付けテーブルのエントリ（すなわち、クラスタ）ごとのセクターの数と、トラックごとのセクター数と、ディスクごとの面の数と、予約セクター又はかげのセクターの数と、物理ドライブ番号と、拡張ブートセクターシグナチャと、ボリューム識別と、ボリュームテーブルとに関する情報をさらに含む。

【００２０】ルートディレクトリ７は、それぞれがファイルの何らかの属性を記述する複数の３２バイトエントリから成るテーブルである。通常、ルートディレクトリ７を構成する各ディレクトリエントリはファイル名と、ファイル拡張と、属性フラグと、ファイルの日付スタンプと、ファイルを構成するクラスタの開始クラスタ番号と、ファイルサイズとを含む。

【００２１】ディスクの各ファイルは１つ又は複数のクラスタから構成されている。ファイル割付テーブル５は、ファイルを構成するクラスタが互いに連係する連鎖の形態をとる記録を含む。典型的なＦＡＴ５はクラスタごとに１つずつある複数の２バイトエントリから成るリストを含む。従来のＦＡＴの中には、ＦＡＴエントリが２バイトを越えるものもある。各ＦＡＴエントリの長さはクラスタの総数によって決まる。ファイルのディレクトリエントリはそのファイルの開始クラスタ番号を含み、オペレーティングシステムはその開始クラスタ番号を使用して、ファイル割付けテーブルをアクセスする。各ＦＡＴエントリはファイルの次のクラスタを指示するポインタである。従って、その最初のアクセスによって検索されるＦＡＴエントリは、ファイルを構成している次のクラスタのクラスタ番号を含む。オペレーティングシステムはその次のクラスタ番号を使用し、ＦＡＴをアクセスしてさらにその次のクラスタ番号を検索し、ＦＡＴ５の中の特別のマーカに到達するまでこのプロセスを継続してゆく。

【００２２】ディスクにおけるファイル構造はツリー形である。ルートディレクトリのエントリはサブディレクトリを指示するポインタを形成することができる。サブディレクトリは入れ子形であっても良い。

【００２３】ある種の従来のパーソナルコンピュータでハードディスクやディスケットを使用することに関連する欠点の１つは、ハードディスクドライブ及びフロッピーディスクドライブが多数の機械的構成要素を伴う相対的に物理的に大形の装置であるという点である。その大きさは、パーソナルコンピュータの他の多くの部品を構成している集積回路の小ささとは対照的である。さらに、相対的に大形である従来の典型的なハードディスクドライブやフロッピーディスクドライブは、持ち運び自在のパーソナルコンピュータをより一層小型にし、持ち運びしやすくしようとする試みには妨げとなる。

【００２４】従来のハードディスクドライブ及びフロッピーディスクドライブに関わるもう１つの欠点は、それらの装置がパーソナルコンピュータの他の部品を構成している集積回路と比較して大量の電力を消費することである。

【００２５】従来のハードディスクドライブ及びフロッピーディスクドライブに関わるさらに別の欠点は、それらの装置が過剰な衝撃や振動又はちり、もしくは他の大気中汚染物質にさらされた場合に故障しやすいことである。

【００２６】従来の別の種類の不揮発性コンピュータメモリはフラッシュ電気的消去可能プログラマブル読取り専用メモリ（「フラッシュＥＥＰＲＯＭ」）である。フラッシュＥＥＰＲＯＭはユーザー側でプログラムすることができ、一度プログラムしてしまえば、消去されるまでデータを保持する。フラッシュＥＥＰＲＯＭを電気的に消去すると、装置のメモリの内容全てが１回の相対的に速い動作で消去される。その後、フラッシュＥＥＰＲＯＭを新たなコードでプログラムすれば良い。

【００２７】ところが、従来のある型のフラッシュＥＥＰＲＯＭには、個々のビットセルをあらかじめ消去せずに論理値０から論理値１に重ね書きすることは不可能であるという欠点がある。従来の１つの型のフラッシュＥＥＰＲＯＭのもう１つの欠点は、大きなブロックの単位で又は装置全体を消去してしまうように消去しなければならないこと、すなわち、論理状態１にリセットしなければならないことである。

【００２８】従来の１つの型のフラッシュＥＥＰＲＯＭの別の欠点は、フラッシュＥＥＰＲＯＭが故障するまでのフラッシュＥＥＰＲＯＭの消去サイクルと書込みサイ

クルの数に限りがあることである。

【００２９】従来のいくつかのフラッシュＥＥＰＲＯＭに関わる重ね書き及び消去についての制限は、場合によっては、パーソナルコンピュータシステムにおけるフラッシュＥＥＰＲＯＭの有用性を限定していた。

【００３０】

【発明が解決しようとする課題】本発明の目的の１つは、ブロック単位でのみ消去可能であり且つあらかじめ消去せずにビットを第１の論理状態から第２の論理状態に重ね書きすることは不可能である不揮発性半導体メモリのファイル構造を提供することである。

【００３１】本発明の別の目的は、ブロック単位でのみ消去可能であり且つあらかじめ消去せずにビットを第１の論理状態から第２の論理状態に重ね書きすることは不可能であり、ファイル構造の保全性及び信頼性を最大にするのを助長する予約ブロックを含むような不揮発性メモリのファイル構造を提供することである。

【００３２】本発明の別の目的は、ブロック単位でのみ消去可能であり且つあらかじめ消去せずにビットを第１の論理状態から第２の論理状態に重ね書きすることは不可能であり、アクティブブロックと、予約ブロックと、ディレクトリブロックとを具備する不揮発性半導体メモリを含むコンピュータシステムを提供することである。

【００３３】

【課題を解決するための手段】ブロック単位でのみ消去可能である不揮発性半導体メモリを説明する。不揮発性半導体メモリの各ビットをあらかじめ消去せずに第１の論理状態から第２の論理状態に重ね書きすることは不可能である。不揮発性半導体メモリの各ビットをあらかじめ消去せずに第２の論理状態から第１の論理状態に重ね書きすることは可能である。不揮発性半導体メモリは、第１のファイルを記憶するアクティブブロックと、第２のファイルを記憶する予約ブロックと、ディレクトリブロックとを具備する。第２のファイルは第１のファイルのコピーである。コピーはアクティブブロックの消去に先立ってクリーンアップ動作中に作成される。ディレクトリブロックは、第１のファイルを識別するためのディレクトリエントリを含む。

【００３４】本発明のその他の目的は、特徴及び利点は添付の図面及び以下の詳細な説明から明らかになるであろう。

【００３５】本発明を添付の図面の各図に例示してあるが、図面は本発明を限定するものではない。また、図中、同じ図中符号は同様の要素を指示する。

【００３６】

【実施例】図２は、１つの好ましいファイルシステム（ファイル構造ともいう）３２を含むパーソナルコンピュータシステム１０を示す。ファイルシステム３２は１つのフラッシュＥＥＰＲＯＭ又はフラッシュメモリアレイ３４を形成する複数のフラッシュＥＥＰＲＯＭを含む。フラッシュＥＥＰＲＯＭは半導体メモリの一種である。ファイル構造３２は、システムＲＡＭ２２に記憶されているソフトウェアファイルシステムドライバ２８をさらに含む。パーソナルコンピュータシステム１０は中央処理装置（「ＣＰＵ」１２）と、バス１８とをさらに含む。

【００３７】以下にさらに詳細に説明する通り、フラッシュメモリアレイ３４は動的メモリ再割当てを可能にするような構造である。フラッシュメモリアレイ３４は可変メモリ構造又はセクター方式ファイル構造のいずれかを有する。フラッシュメモリアレイ３４と関連して連係リストディレクトリ又はディスクアナログディレクトリのいずれかを使用する。ディスクアナログディレクトリはバイト消去可能ＥＥＰＲＯＭなどの個別のメモリに常駐している。連係リストディレクトリはフラッシュメモリ、又はバイト消去可能ＥＥＰＲＯＭなどの別個のメモリに常駐している。

【００３８】システムＲＡＭ２２はアプリケーションプログラム２４を記憶するためのスペースを含む。本発明の一実施例では、システムＲＡＭ２２は１メガバイトの大きさである。システムＲＡＭ２２は、他のアプリケーションプログラムを記憶するためと、データのためのスペースをさらに含む。本発明の一実施例においては、システムＲＡＭ２２はコンピュータシステム１０に対して１つのオペレーティングシステム２６を含む。別の実施例では、システムＲＡＭ２２はさらに２つ以上のオペレーティングシステムを含む。

【００３９】システムＲＡＭ２２はファイルシステムドライバソフトウェア２８をさらに含む。ファイルシステムドライバソフトウェア２８はフラッシュメモリアレイ３４のファイル構造を規定する。パーソナルコンピュータシステム１０では、中央処理装置１２は、フラッシュメモリアレイ３４に関わるあらゆるファイルシステム管理ユーティリティを処理するために、ファイルシステムドライバソフトウェア２８のフォアグラウンドタスクソフトウェアルーチンを実行する。ファイルシステムドライバソフトウェア２８と、フラッシュメモリアレイのファイル構造については以下にさらに詳細に論じる。

【００４０】システムＲＡＭ２２がアプリケーションプログラム２４，オペレーティングシステム２６及びファイルシステムドライバソフトウェア２８を記憶するのみならず、アプリケーションプログラム２４，オペレーティングシステム２６及びファイルシステムドライバソフトウェア２８は、パーソナルコンピュータシステム１０がオフ又はオンである間に、フラッシュメモリアレイ３４にも記憶される。フラッシュメモリアレイ３４は２つ以上のアプリケーションプログラムと、２つ以上のオペレーティングシステムと、２つ以上のファイルシステムドライバソフトウェアとを記憶できる。フラッシュメモリアレイ３４はデータファイルと、ディレクトリ情報と

をさらに記憶する。ウォームブート又はコールドブートのいずれかの時点で、ファイルシステムドライバソフトウェア２８はフラッシュメモリアレイ３４からシステムＲＡＭ２２へロードされる。要するに、フラッシュメモリアレイはハードディスクの代わりとなる大容量記憶装置として動作する。

【００４１】パーソナルコンピュータシステム１０は、自身に関わるＢＩＯＳソフトウェアを記憶するＲＯＭ１４をさらに含む。本発明の一実施例では、ＲＯＭ１４に記憶されるＢＩＯＳは、それぞれ２キロバイトの追加ＢＩＯＳ　ＲＯＭ拡張部を伴う６４キロバイトＢＩＯＳを含む。ファイルシステムドライバソフトウェア２８は、実行に先立って、ＲＯＭ１４ではなくフラッシュメモリアレイ３４に記憶される。

【００４２】コンピュータシステム１０のパワーアップ後、ＲＯＭ１４のＢＩＯＳプログラムを実行する時点で、ファイルシステムドライバソフトウェア２８とオペレーティングシステム２６はフラッシュメモリアレイ３４からバス１８を介してシステムＲＡＭ２へロードされる。そこで、コンピュータシステム１０のユーザーはオペレーティングシステム２６を使用して、アプリケーションプログラム２４をシステムＲＡＭ２２にロードする。コンピュータシステム１０のユーザーは、さらに、データ、他のアプリケーションプログラム及び他のオペレーティングシステムをシステムＲＡＭ２２にロードすることができる。

【００４３】本発明の一実施例では、フラッシュメモリアレイ３４はブロックごとに消去され、各ブロックは多数のビットから構成されている。別の実施例においては、フラッシュメモリアレイ３４はことごとく消去される。

【００４４】フラッシュメモリは、消去に関して、従来の電気的消去可能プログラマブル読取り専用メモリ（「ＥＥＰＲＯＭ」）とは異なる。従来のＥＥＰＲＯＭは個々のバイト消去制御のためにセレクトトランジスタを使用する。それに対し、フラッシュメモリは個々のトランジスタセルによってはるかに高い密度を達成する。消去モードの間に、ブロックの中のあらゆるメモリセル又はチップ全体にあるあらゆるメモリセルのソースに同時に高電圧を供給すると、その結果、アレイ全体又はブロック全体が消去されることになる。

【００４５】フラッシュメモリアレイ３４の場合、論理値「１」は、１つのビットセルと関連しているフローティングゲートに、あったとしてもごくわずかの電子しか記憶されていないということを意味する。論理値「０」は、ビットセルと関連するフローティングゲートに多数の電子が記憶されていることを意味する。フラッシュメモリアレイ２３が消去された後、フラッシュメモリアレイ２３の各ビットセルには論理値１が記憶されている。フラッシュメモリアレイ２３の個々のビットセルを、あらかじめ消去せずに、論理値０から論理値１に重ね書きすることは不可能である。ただし、フラッシュメモリアレイ２３の個々のビットセルを論理値１から論理値０に重ね書きすることは、それが、単に、本来、消去状態と関連する数の電子を含んでいるフローティングゲートに電子を追加することのみを伴うのであれば可能である。

【００４６】フラッシュメモリアレイ３４は（１）　一度に１ビットずつ、（２）　一度に１語ずつ又は（３）　一度に語群単位のいずれかの方法でプログラム可能である。１つの語は単一のメモリシステムアドレス又は単一の装置と関連する複数のメモリビットから構成される。一度にプログラムされる語群はフラッシュメモリアレイ３４の１つの消去ブロックと同じ大きさであることができる。フラッシュメモリに関するプログラム動作を書込み動作ともいう。

【００４７】先の述べた通り、フラッシュメモリアレイ３４の各ビットを、あらかじめ消去せずに、論理状態０から論理状態１に重ね書きすることは不可能である。論理値０から論理値１への重ね書きに先立ってこのように消去が必要であるために、フラッシュメモリと関連して１つの機能動作が導入される。

【００４８】フラッシュメモリアレイ３４は実行可能コードと、実行不能データの双方を記憶する。以下の詳細な説明の中では、総称用語としての「データ」は（１）　実行不能データのみを表わすか、（２）　実行可能コードと実行不能データの双方を表わすために使用される。

【００４９】フラッシュメモリアレイ３４に関連する読取り動作は、他の読取り専用メモリ装置に関連する読取り動作とごく類似している。一実施例では、フラッシュメモリアレイ３４の読取り動作は１３５ナノ秒を要する。ところが、フラッシュメモリアレイの書込み動作と消去動作はそれよりはるかに遅い。本発明の一実施例においては、フラッシュメモリアレイ３４の１つのブロックの消去時間は１秒である。フラッシュメモリアレイ３４の１つの語の書込み動作は１０マイクロ秒を要する。従って、フラッシュメモリアレイ３４に関連する読取り動作と、書込み・消去動作とは非対称の関係にあり、書込み動作と消去動作は読取り動作より著しく遅い。

【００５０】さらに、フラッシュメモリアレイ３４の各フラッシュメモリは消去並びに書込み（すなわち、プログラム）動作に関して限られた耐久寿命を有する。たとえば、一実施例では、フラッシュメモリアレイ３４は、故障し始めるか又は消去／書込み性能の劣化を示すまでに少なくとも１００００消去／プログラムサイクルに耐えることができる。従って、本発明の一実施例においては、フラッシュメモリアレイ３４が完了できる消去／書込みサイクルの数に限りがある。これに対し、フラッシュメモリアレイ３４が読取りを実行できる回数については、そのような最長寿命は存在していない。

【００５１】本発明の一実施例では、フラッシュメモリアレイ３４を構成している各フラッシュメモリは、それぞれのメモリ装置の中に複数の消去ブロックを有している。その実施例の場合、隣接するブロックを巡る消去／プログラミングの循環には制限がある。隣接ブロックとは、共通の行接続又は共通の列接続を共有しているブロックである。隣接ブロックを巡る循環の制限は、電気的データ妨害状態を阻止するために設けられる。

【００５２】循環の頻度の多いブロックをホットブロックといい、循環の起こらないか又は循環の頻度が少ないブロックをコールドブロックという。重要なパラメータは、隣接ブロックとそれぞれ関連するサイクルの数の差である。ホットブロックは、最終的には、フラッシュメモリアレイ３４のコールドブロックではデータが悪影響を受けてしまうようなサイクルカウントに達することもある。このホットブロック／コールドブロック相互作用を回避するための１つの方法は、フラッシュメモリアレイ３４のホットブロックにサイクルの制限を課すことである。ホットブロック／コールドブロック相互作用の悪影響を最小限に抑える別の方法は、フラッシュメモリアレイ３４のコールドブロックを周期的に消去し、再プログラムするというものである。コールドブロックの周期的な消去と再プログラミングをコールドブロックの再生という。

【００５３】以下にさらに詳細に説明する通り、本発明の好ましいファイル構造は、フラッシュメモリアレイ３４を構成するフラッシュＥＥＰＲＯＭの先に挙げたような機能上の特性を考慮に入れている。

【００５４】図３は、１つの好ましいファイル構造５８を有するパーソナルコンピュータシステム４０を示す。パーソナルコンピュータシステム４０は中央処理装置１２と、ＲＯＭ４２とを含む。ＲＯＭ４２はＢＩＯＳソフトウェア４３と、ＢＩＯＳ拡張ソフトウェア４４とを記憶している。パーソナルコンピュータシステム４０はバス４８と、システムＲＡＭ５２とをさらに含む。システムＲＡＭ５２はアプリケーションプログラム２４と、オペレーティングシステム２６とを記憶している。パーソナルコンピュータシステム４０はＲＡＭバッファ６２と、フラッシュメモリアレイ６４とをさらに含む。

【００５５】ＲＡＭバッファ６２はフラッシュメモリアレイ６４とシステムＲＡＭ５２との間のファイルの転送に関して緩衝を実行する。さらに、ＲＡＭバッファ６２はフラッシュメモリアレイ６４に記憶されているファイルについてのディレクトリの構成又は更新に関しても緩衝を実行する。

【００５６】１つ又は複数のフラッシュＥＥＰＲＯＭがフラッシュメモリアレイ６４を構成している。フラッシュメモリアレイ６４はアプリケーションプログラム２４と、オペレーティングシステム２６と、データと、他のアプリケーションプログラムと、他のオペレーティングシステムを記憶する。オペレーティングシステム２６、アプリケーションプログラム２４及び他のプログラムとデータはフラッシュメモリアレイ６４からバス４８を介してシステムＲＡＭ５２へ転送可能である。

【００５７】ファイル構造５８はフラッシュメモリアレイ６４と、ＲＡＭバッファ６２と、ＲＯＭ　ＢＩＯＳ拡張ソフトウェア４４と、ＲＯＭ　ＢＩＯＳソフトウェア４３の一部とを含む。

【００５８】フラッシュメモリアレイ６４のファイル構造を規定するファイルシステムドライバソフトウェアはＲＯＭ４２にあるＲＯＭ　ＢＩＯＳソフトウェア４３及びＲＯＭ　ＢＩＯＳ拡張ソフトウェア４４の一部を形成している。図３に示す実施例では、ファイルシステムドライバはパーソナルコンピュータシステム４０の一部であり且つそれと一体である。ＢＩＯＳソフトウェア４３とＢＩＯＳ拡張ソフトウェア４４は、直接マッピング方式又はページング方式のメモリブロックである。従って、図３に示す実施例においては、ファイルシステムドライバソフトウェアはシステムＲＡＭ５２にも、フラッシュメモリアレイ６４にも記憶されない。

【００５９】図４は、さらに別の好ましいファイル構造９４を有するパーソナルコンピュータシステム７０を示す。パーソナルコンピュータシステムは中央処理装置１２と、システムＲＡＭ７２とを含む。好ましい一実施例では、システムＲＡＭ７２はアプリケーションプログラム２４と、オペレーティングシステム２６とを記憶している。パーソナルコンピュータシステム７０はバス８８と、ＲＯＭ７８とをさらに含む。ＲＯＭ７８はＢＩＯＳソフトウェア７９と、ＢＩＯＳ拡張ソフトウェア８０とを記憶している。

【００６０】パーソナルコンピュータシステム７０は制御装置９２と、フラッシュメモリアレイ９６とをさらに含む。フラッシュメモリアレイ９６は複数のフラッシュＥＥＰＲＯＭから構成されている。フラッシュメモリアレイ９６はアプリケーションプログラム２４と、オペレーティングシステム２６と、データと、他のアプリケーションプログラムと、他のオペレーティングシステムとを記憶し、それらをバス８８を介してシステムＲＡＭ７２にロードすることができる。ファイル構造９４は制御装置９２と、フラッシュメモリアレイ９６と、ＲＯＭＢＩＯＳ拡張ソフトウェア８０と、ＲＯＭ　ＢＩＯＳソフトウェア７９の一部とから構成されている。

【００６１】図４に示すパーソナルコンピュータシステム７０は、フラッシュメモリアレイ９６に関して完全ハードウェア駆動形ファイル構造を有する。制御装置９２は、フラッシュメモリアレイ９６のファイル構造を規定し且つ制御するための特定専用装置である。制御装置９２は、制御を監督するマイクロコントローラ１００を含む。

【００６２】制御装置９２は制御論理１０６をさらに含

む。本発明の好ましい一実施例においては、制御論理１０６はプログラマブル論理アレイ（「ＰＬＡ」）から構成される。制御論理１０６は、制御装置９２の動作を制御するデジタル論理を含む回路である。

【００６３】一実施例では、制御装置９２は、ファイルシステムドライバソフトウェアを記憶するＲＯＭ９８を含む。ファイルシステムドライバソフトウェアはマイクロコントローラ１００により実行可能であり、フラッシュメモリアレイ９６のファイル構造を規定する。

【００６４】別の実施例によれば、フラッシュメモリアレイ９６のファイル構造を規定する実行可能なファイルシステムドライバソフトウェアは、ＲＯＭ　ＢＩＯＳソフトウェア７９の一部及びＲＯＭ　ＢＩＯＳ拡張ソフトウェア８０を形成する。

【００６５】制御装置９２はバッファＲＡＭ１０２を含む。バッファＲＡＭ１０２はファイルの転送と、ディレクトリの作成及び更新とに関して緩衝を実行する。

【００６６】制御装置９２はバッファ／マルチプレクサ１０４をさらに含む。バッファ／マルチプレクサ１０４はファイルの転送に関して緩衝を実行する。バッファ／マルチプレクサ１０４は、さらに、フラッシュメモリアレイ９６に出入りするデータを多重化する。

【００６７】論理バッファ／マルチプレクサ回路１０４と関連する論理はいずれかの標準システムバスインタフェースに準拠していれば良い。たとえば、バッファ／マルチプレクサ１０４と関連する論理はＰＣ　ＸＴ，ＰＣ　ＡＴ（すなわち、ＩＤＥ－工業規格アーキテクチャ）、ＥＩＳＡ（すなわち、、拡張工業規格アーキテクチャ）、ＭＣＡ（すなわち、マイクロチャネルアーキテクチャ）、ＶＭＥ（すなわち、仮想機械環境）及びマルチバスなどの規格のうち１つ又は２つ以上に準拠していれば良い。

【００６８】本発明の一実施例においては、制御装置９２はパーソナルコンピュータシステム７０の中に含まれている。たとえば、制御装置９２はパーソナルコンピュータシステム７０内部のシステムボード（図示せず）に位置していることも可能であろう。

【００６９】別の実施例では、制御装置９２は、パーソナルコンピュータシステム７０の外にある又はパーソナルコンピュータシステム７０の拡張用スロット（図示せず）の中にある入出力（「Ｉ／Ｏ」）装置であっても良い。その実施例の場合、バッファ／マルチプレクサ１０４と関連する論理はいずれかの標準Ｉ／Ｏインタフェースに準拠していれば良い。たとえば、バッファ／マルチプレクサ１０４はＩＤＥ，ＳＴ５０６，ＳＣＳＩ（すなわち、小型コンピュータシステムインタフェース）及びＳＡ４００（すなわち、フロッピーディスク規格）などのＩ／Ｏインタフェースの１つ又は２つ以上に準拠していれば良い。従って、その実施例では、パーソナルコンピュータシステム７０はそのような標準Ｉ／Ｏインタフェースの中の１つを介して制御装置９２と通信すると考えられる。

【００７０】図３のパーソナルコンピュータシステム４０と、図４のパーソナルコンピュータシステム７０は、対応するフラッシュメモリアレイのファイル構造を制御するための専用のハードウェアをそれぞれ含んでいる。パーソナルコンピュータシステム４０の場合、その追加ハードウェアはＲＡＭバッファ６２である。パーソナルコンピュータシステム７０の場合には、その追加ハードウェアは制御装置９２である。図３のＲＡＭバッファ６２と、図４の制御装置９２は、ファイルシステム制御コードをシステムＲＡＭ５２，フラッシュメモリアレイ６４，システムＲＡＭ７２又はフラッシュメモリアレイ９６にそれぞれ記憶する必要をなくすための手段を構成している。

【００７１】ＲＡＭバッファ６２と制御装置９２は、図３のシステムＲＡＭ５２と、図４のシステムＲＡＭ７２がそれぞれファイル構造５８，９４に対するデータバッファとして動作する必要がないように、それぞれデータを緩衝する働きをする。さらに、パーソナルコンピュータシステム４０にＲＡＭバッファ６２を含め、また、パーソナルコンピュータシステム７０に制御装置９２を含めることは、システムＲＡＭ５２及びシステムＲＡＭ７２がそれぞれフラッシュメモリアレイ６４及び９６のファイル構造に関するスクラッチパッド領域として動作する必要がないということを意味している。

【００７２】図３のＲＡＭバッファ６２と、図４の制御装置９２とを設けたため、ファイル構造５８とファイル構造９６を、それぞれ、対応するパーソナルコンピュータシステムに関する基本システムハードウェアの一部であるブート可能記憶領域を含むものとして認識することができる。その結果、ＲＡＭバッファ６２と制御装置９２はそれぞれ対応するパーソナルコンピュータシステムのフラッシュメモリアレイのファイル構造の総体的な性能を向上させる働きをする。

【００７３】図２～図４に示すパーソナルコンピュータシステム１０，４０及び７０は、構成の上で異なっているので、コンピュータの特性も異なる結果になる。図２のコンピュータシステム１０では、ファイルシステムドライバソフトウェア２８は中央処理装置１２により、フラッシュメモリアレイ３４を管理するファイル管理ユーティリティの全てを処理するフォアグラウンドタスクソフトウェアルーチンとして実行される。ファイルシステムドライバソフトウェア２８はシステムＲＡＭ２２の一部を占めるが、システムＲＡＭ２２の大きさは限定されても良いであろう。たとえば、パーソナルコンピュータシステム１０のオペレーティングシステム２６がＭＳ－ＤＯＳである場合、システムＲＡＭ２２を１メガバイトの「実モード」スペースに限定しても良い。

【００７４】図３のパーソナルコンピュータシステム４

０の場合、フラッシュメモリアレイ６４のファイル構造を規定するはファイルシステムドライバソフトウェアは、ＲＯＭ４２のＲＡＭ　ＢＩＯＳソフトウェア４３の一部と、ＲＯＭ　ＢＩＯＳ拡張ソフトウェア４４を形成する。しかしながら、ＲＯＭ４２の記憶スペースの量を限定しても良い。たとえば、ＭＳ－ＤＯＳと互換性をもつパーソナルコンピュータによっては、全ての入出力機能をＲＯＭの６４キロバイトＢＩＯＳ部分に含まれているファームウェアによって駆動するものがあり、その場合のＲＯＭ　ＢＩＯＳ拡張ソフトウェアは２キロバイトずつと少なくなっている。ＲＯＭ４２のサイズが小さいならば、ファイルシステムドライバソフトウェアはＲＯＭ４２で利用可能なスペースを越えても良いであろう。

【００７５】これに対し、コンピュータシステム７０のハードウェア制御装置９２はファイルシステムドライバを記憶し、これにより、フラッシュメモリアレイ９６，ＲＯＭ７８及びシステムＲＡＭ７２に関するメモリ要件は緩和される。さらに、ハードウェア制御装置９２はファイルシステム制御指令を簡略化する。たとえば、ハードウェア制御装置９２は、ファイルシステム制御指令を、他のいずれかのメモリスペースにあるコードを参照するラベルによりソフトウェアアクセス可能である高レベル手順呼出しに変換する。本発明の一実施例においては、簡略化されたファイルシステム制御指令又はファイルドライバ制御指令は従来のＢＩＯＳ型制御コードとしてハードウェア制御装置９２のＲＯＭ８９に記憶される。これにより、ソリッドステートファイルシステム９４を基本ブート可能装置として認識することができる。その結果、ファイルシステムの呼出しは簡単になる。ファイルシステムを呼出すたびに、ファイルシステムドライバ（ファイルシステム制御コードともいう）が実行されることになる。

【００７６】一実施例では、図４のパーソナルコンピュータシステム７０の場合の各ファイルシステム呼出しは、ＲＯＭ７８に記憶されている主ＢＩＯＳ及び直接マッピング方式ＢＩＯＳ拡張ソフトウェアからＣＰＵ１２が命令を取出すことによって、直接実行される。別の実施例では、各ファイルシステム呼出しは、ＲＯＭ９８に記憶されているＢＩＯＳ拡張メモリブロックにあるメモリページからＣＰＵ１２により実行される。さらに別の実施例においては、各ファイルシステム呼出しはハードウェア制御装置９２のＲＯＭ９８に記憶されているファイルシステムドライバからマイクロコントローラ１００により実行される。

【００７７】好ましい実施例では、ハードウェア制御装置９２は各ファイル関連タスクをホストＣＰＵ１２からの単一の指令に短縮し、これがＣＰＵ１２のオフローディングを増加させるのを助ける。制御装置９２は入力データを局所ＲＡＭバッファスペース１０２に保持することにより、入力データをＲＡＭ型メモリ速度を受入れる。ハードウェア制御装置９２は与えられたタスクをそれぞれ自動的に完了するが、一度に１つの動作しか処理しない。

【００７８】さらに、制御装置９２はホストＣＰＵ１２に即時データアクセスのためにいずれかのファイルシステムタスクへの割込みを実行させる。これは、そうでなければアクセス不能ブロックにあると思われる有効ファイル情報の追加ＲＡＭ緩衝を要求する。アクセス不能ブロックは、書込み動作又は消去動作に関連するブロックである。制御装置９２は、割込みに引続いて元のタスクを再開させる。本発明の一実施例においては、ハードウェア制御装置９２は所定の数の同時書込みタスクをキューアップする。

【００７９】さらに、一実施例では、制御装置９２は、利用可能なフリースペースとホット／コールドブロックサイクルの不均衡を求めて、主フラッシュメモリ記憶装置アレイ９６を絶えず監視している。ハードウェア制御装置９２は、さらに、必要に応じてフラッシュメモリアレイ９６に関する再配分動作を自動的に開始させる。再配分動作はフラッシュメモリアレイ９６のブロックの間に循環を均等化させる。再配分動作は必要に応じてハードウェア制御装置９２により完全バックグラウンドタスク方式で自動的に開始される。

【００８０】ハードウェア制御装置９２は、フラッシュメモリアレイ９６における記憶のためのスペースを解放するために、フラッシュメモリアレイ９６に関するクリーンアップ動作をも自動的に開始する。クリーンアップ動作はハードウェア制御装置９２により必要に応じて開始され、完全バックグラウンドタスク方式で実行される。クリーンアップ動作については以下にさらに詳細に説明する。

【００８１】ファイルシステム制御のためのハードウェア制御装置９２が利用可能なフリースペースと、ホット／コールドブロックサイクルの不均衡とを求めてフラッシュメモリアレイ９６を監視し続けると共に、制御装置９２がクリーンアップ動作又は再配分動作を自動的に開始させるには、フラッシュメモリアレイ（９６）は

（１）　パーソナルコンピュータシステム７０の内部の記憶装置として常駐しているか、あるいは、（２）　フラッシュメモリアレイ９６が取外し自在であればパーソナルコンピュータシステム７０に対して確実に物理的にインタロックしているべきである。フラッシュメモリアレイ９６が取外し自在であるこの別の実施例では、パーソナルコンピュータシステム７０は、フラッシュメモリアレイ９６に関わるバックグラウンドタスクが発生しているときに点灯するインジケータとしての発光ダイオード（「ＬＥＤ」）を含む。パーソナルコンピュータシステム７０のユーザーは、ＬＥＤが点灯しているとき、従って、バックグラウンドタスクが発生しているときにはフラッシュメモリアレイ９６を取外してはならないとい

う命令を受ける。

【００８２】好ましい実施例では、主フラッシュメモリアレイに対するインタフェース回路を（１）　入出力（「Ｉ／Ｏ」）マッピング、（２）　ページング形メモリマッピング又は（３）　直接メモリマッピングすることができる。図５は、インタフェース回路に関するＩ／Ｏマッピング方式を示す。

【００８３】図２～図４に示す実施例の場合、固定ディスクドライブ又はフロッピーディスクドライブに代わるものとしてフラッシュメモリアレイ３４，６４及び９６を使用できるであろうということを理解すべきである。

【００８４】図５に示すＩ／Ｏマッピング方式は、フラッシュメモリアレイ８１及び局所バッファＲＡＭ８３に対してマッピングされる直列転送Ｉ／Ｏポートから成るＩ／Ｏプレーン８５を使用する。図５に示すＩ／Ｏマッピング方式は主メモリプレーン（システムＲＡＭともいう）の代わりに別個にＩ／Ｏメモリプレーン８５を使用するので、図５に示すＩ／Ｏマッピング方式はホストコンピュータの主メモリスペースを全く消費しない。すなわち、Ｉ／Ｏマッピング方式はホストコンピュータのシステムＲＡＭを全く消費しないのである。

【００８５】ところが、直列転送Ｉ／Ｏポート８５に結合するＩ／Ｏ装置は、それらのＩ／Ｏ装置（図示せず）がランダムアクセス装置ではなく、直列装置であると、中央処理装置による直接実行を支援することができない。それらのＩ／Ｏ装置は実行に備えてコードファイルをシステムＲＡＭにダウンロードしなければならない。

【００８６】図６は、ページング形マップフラッシュメモリアレイ１０５に対するページング形メモリマッピングインタフェースを示す。図６は、主メモリプレーン８９を示す。主メモリプレーン８９は主システムＲＡＭ９９と、ページングウィンドウ９７と、ＢＩＯＳ／ＲＡＭ拡張ソフトウェア９５と、ＢＩＯＳ９３と、拡張システムＲＡＭ９１とを含む。

【００８７】本発明の一実施例においては、主メモリプレーン８９は１メガバイトのアドレス範囲制約１０１を有する。主メモリプレーン８９について１メガバイトのアドレス範囲制約があるということは、１メガバイトのアドレス範囲制約１０１を越える主メモリプレーン８９の部分をアドレッシングするためにＣＰＵ又はマイクロプロセッサの保護モードを使用することを意味する。言いかえれば、ＣＰＵの保護モードを使用して、主メモリプレーン８９の拡張システムＲＡＭ領域９１をアドレッシングする。

【００８８】初期の世代の典型的な低性能マイクロプロセッサは保護モードを含んでいなかった。さらに、パーソナルコンピュータシステムの初期の世代の典型的な低性能オペレーティングシステムは、保護モードアドレッシングを処理する能力を有していなかった。従って、そのような性能の低いマイクロプロセッサやオペレーティングシステムは１メガバイトのアドレスを越える領域をアドレッシングできなかった。それに反して、後の世代の高性能のマイクロプロセッサやオペレーティングシステムの中には、１メガバイトアドレスを越える領域でもアドレッシングできるものがある。

【００８９】図６では、フラッシュメモリアレイ１０５は小形ページ１０７，１０９及び１１１から構成されており、局所バッファＲＡＭ１１３に結合している。本発明の別の実施例においては、フラッシュメモリアレイ１０５をどのような数のページからも構成できると考えられる。

【００９０】図６のページングウィンドウ９７は、ホストの中央処理装置からフラッシュメモリアレイ１０５のページ１０７，１０９及び１１１への直接アクセスを可能にするページング形メモリマッピングインタフェースである。各ページは必要に応じて異なるフラッシュメモリアレイ１０５のセグメントを指示するように変更される。

【００９１】図６は、装置ドライバを拡張するものとして使用されるページング形ファームウェア１０３をさらに示す。ページング形ファームウェア１０３はページ１１５，１１７，１１９及び１２１を含む。本発明の一実施例では、ページングウィンドウインタフェース９７を使用して、ページング形ファームウェア１０３を指示する。ページングウィンドウ９７によってページング形ファームウェア１０３を指示すれば、さらに大きなファームウェア記憶スペース１０３への拡張が可能になる。これにより、単一の（通常は２キロバイトの）ＢＩＯＳ／ＲＯＭ拡張記憶場所９５を越える追加の記憶スペースがコンピュータシステムに与えられることになる。

【００９２】図７は、主フラッシュメモリアレイへの直接メモリマッピング形インタフェースを示す。図７では、主メモリプレーン１３５は、主システムＲＡＭ１３７と、ＢＩＯＳ／ＲＯＭ拡張ソフトウェア１３９と、ＢＩＯＳ１４１と、拡張システムＲＡＭ１４５とに加えて、フラッシュメモリアレイマップ１４９と、局所バッファＲＡＭ１４７とを含む。

【００９３】図７に示す主メモリプレーン１３５は１メガバイトのアドレス範囲制約１４３を有する。フラッシュメモリアレイマップ１４９と、局所バッファＲＡＭマップ１４７はメモリプレーン１３５の１メガバイトのアドレス範囲制約１４３を越えたところにある。すなわち、フラッシュメモリアレイマップ１４９と局所バッファＲＡＭマップ１４７をアドレッシングするときには、ＣＰＵ又はマイクロプロセッサの保護モードを使用することになる。これに対し、ＣＰＵは主メモリプレーン１３５の主システムＲＡＭ１３７と、ＢＩＯＳ／ＲＯＭ拡張ソフトウェア１３９と、ＢＩＯＳ１４１については保護モードを使用せずにアドレッシングできる。

【００９４】フラッシュメモリアレイマップ１４９はパ

ーソナルコンピュータシステムのフラッシュメモリアレイに対し直接マッピングする。従って、フラッシュメモリアレイマップ１４９の大きさはパーソナルコンピュータシステムのフラッシュメモリ記憶アレイと等しい。

【００９５】さらに、局所バッファＲＡＭマップ１４７はパーソナルコンピュータシステムの局所バッファＲＡＭに対し直接マッピングされている。そのため、局所バッファＲＡＭマップ１４７の大きさはパーソナルコンピュータシステムの局所バッファＲＡＭと等しい。

【００９６】このように、直接メモリマッピング方式は、大量の主メモリスペースを使用する。さらに、直接メモリマッピング方式では、１メガバイトのアドレス範囲制約１４３を越える主メモリプレーンの部分をアドレッシングすることができるＣＰＵとオペレーティングシステムを必要とする。

【００９７】直接マッピング方式によれば、フラッシュメモリアレイの全てから直接コード実行が可能である。これに対し、ページング形マッピング方式ではフラッシュメモリアレイの一部からの直接コード実行が可能である。

【００９８】直接マッピング方式は１メガバイトの制約を越えてアドレッシングすることができる。ＣＰＵとオペレーティングシステムを要求する。これに対し、ページング形マッピング方式は、１メガバイトの制約を越えてアドレッシングすることが不可能であるＣＰＵ及びオペレーティングシステムと共に動作する。

【００９９】図８，図１１，図１２及び図１４から図１６は、パーソナルコンピュータシステムのフラッシュメモリＥＰＲＯＭアレイについて好ましいファイル構造の例を示している。特に、図８，図１１，図１２及び図１４から図１６は、図２から図４のパーソナルコンピュータシステム１０，４０及び７０のフラッシュメモリアレイ３４，６４及び９６について好ましいファイル構造をそれぞれ示している。図８，図１１及び図１２に示すファイル構造は可変ファイル構造であり、図１４から図１６に示すファイル構造はセクター方式ファイル構造である。図８及び図１４に示すファイル構造は、それぞれ、動的ファイルディレクトリを含み、これはフラッシュメモリアレイの中にある。図１１，図１２，図１５及び図１６に示すファイル構造は、それぞれ、別個のＥＥＰＲＯＭ又はスタティックＲＡＭ（「ＳＲＡＭ」）に記憶されるファイルディレクトリを含む。図８，図１１，図１４及び図１５に示すファイル構造は連係リストディレクトリエントリを含む。図１２及び図１６に示すファイル構造は、それぞれ、従来のハードディスク又は固定ディスクに使用されていたディレクトリに類似するファイルディレクトリを含む。可変ファイル、セクター方式ファイル、動的ディレクトリ、別個に記憶されるディレクトリ、連係リストディレクトリ及びディスク類似のディレクトリについては、以下にさらに詳細に説明する。図８、図１１，図１２及び図１４から図１６に示すファイル構造は、それぞれ、クリーンアップ動作の直後の状態にある。

【０１００】図８は、フラッシュメモリアレイ１１２のファイル構造１１０を示す。ファイル構造１１０は可変サイズファイル構造である。可変サイズファイル構造を可変ファイル構造ともいう。

【０１０１】ファイル構造１１０のような可変ファイル構造においては、パーソナルコンピュータのＣＰＵのプログラムカウンタがコード又はデータを通してステップできるように、コード又はデータを間断なく記憶してゆく。従って、可変ファイル構造は現場実行を可能にする。現場実行とは、ＣＰＵがＲＡＭを必要とせずにフラッシュメモリアレイから直接にコードを実行できることを意味する。現場実行を直接コード実行又は現場コード実行ともいう。

【０１０２】ファイル構造１１０は可変ファイル構造であるため、直接現場実行動作に備えてコードファイルを大きさに関係なく間断なく記憶することができる。現場実行動作によって、実行に備えてＲＡＭ型メモリにダウンロードするための時間を費やさなくて済むようになる。従って、フラッシュメモリに可変ファイル構造を使用すると、パーソナルコンピュータシステムにおけるＲＡＭ要件は緩和される。その結果、このようなファイル構造を使用しない場合に必要になると思われる余分なＲＡＭに費やされる分だけ、システムのパワーを節約できる。

【０１０３】ファイル構造１１０を適正に定様式化した場合、主アレイにあるヘッダを除いて、ホットブロック又はコールドブロックは存在していない。すなわち、電気的データ妨害状態を防止するための隣接ブロックの消去／再プログラミング循環を追跡する必要はないのである。

【０１０４】一方、フラッシュメモリアレイの可変ファイル構造は従来の典型的なディスクドライブに見られるセクターに基づくファイル構造とはかなり異なっている。パーソナルコンピュータの従来の典型的なオペレーティングシステムは、ディスクセクター方式ファイル構造と組合わせた場合にのみ動作するように設計されている。従って、可変ファイル構造と共に動作するためには、現在の典型的な高レベルオペレーティングシステムを修正するか又は追加しなければならないと考えられる。

【０１０５】さらに、可変ファイル構造の場合、記憶スペースの再割付けのためのファイルクリーンアップ動作（以下にさらに詳細に説明する）はセクター方式ファイル構造より効率が悪く且つより頻繁である。従って、可変ファイル構造では、クリーンアップ動作はより長いバックグラウンド時間を要する。さらに、可変ファイル構造はより多くの総合フラッシュメモリ消去／書込み循環

を必要とする。

【０１０６】フラッシュメモリアレイ１１２は、コード又はデータを記憶する多数のデータブロックを含む。図８は、クリーンアップ動作直後の状態にあるフラッシュメモリアレイ１１２のファイル構造の１例を示す。

【０１０７】フラッシュメモリアレイ１１２のブロック１１４，１１６及び１１８は、予約ブロック又は予約済みブロックとも呼ばれる「予約となるべき」ブロックである。予約ブロックは、クリーンアップ動作中に一時ファイルバックアップを実行するブロックである。クリーンアップ動作は再割付けのために使用される。クリーンアップ動作は、まず、最終的に消去されるブロックから現在活動中である全てのファイルを取外すことから成る。各フラッシュメモリブロックは全て一度に消去できる。従って、予約ブロック１１４，１１６及び１１８はクリーンアップ動作中の消去／再割当てのために利用可能なブロックである。このように、予約ブロック１１４，１１６及び１１８は一時ファイルバックアップを実行する。

【０１０８】一実施例では、予約ブロックの中の少なくとも１つを予備予約ブロックとして使用する。この予備ブロックは、クリーンアップ動作のために確実に１つの予約ブロックを利用できることを保証するのを助け、それにより、ファイルシステムの信頼性を向上させるのを助ける。

【０１０９】図８に示すように、ブロック１２０は空きとなるべきブロックである。

【０１１０】ブロック１２２，１２４，１２６及び１２８は空きブロックである。空きデータブロック１２２，１２４，１２６及び１２８は、コード及びデータを記憶するために使用可能なブロックである。図８の空きブロック１２２，１２４，１２６及び１２８はアクティブブロックではなく、コード又はデータを記憶するために使用できるブロックである

【０１１１】パーソナルコンピュータシステムの動作中、空きブロックにデータ又はコードが導入される。データブロック１３０は部分空きブロックである。ブロック１３０の一部はアクティブコード又はアクティブデータを含む。ブロック１３０の残る部分は一部が空いており、後にコード又はデータを記憶するためにその空き部分を利用できる。

【０１１２】空きブロックや部分空きブロックがコードとデータで充填されてゆくにつれて、データ又はコードを含まないままであるブロックの数は減少する。使用可能な空きブロックの数が予約ブロックの許容数に達すると、クリーンアップ動作が始まる。

【０１１３】ブロック１３２，１３６，１３８，１４０，１４２及び１４４は、ブロック１３０の一部と共に、アクティブ連続可変サイズファイル及び削除連続可変サイズファイルを構成する。一実施例では、ブロック１３２，１３６，１３８，１４０，１４２及び１４４はブロック１３０の一部と共にアクティブサブディレクトリ及び削除サブディレクトリをさらに含む。

【０１１４】ファイル１５０，１６４，１６６，１６８，１７０，１７２及び１７４はブロック１３２，１３６，１３８，１４０，１４２及び１４４の中の削除ファイルである。図８に示す削除ファイル１５０，１６４，１６６，１６８，１７０，１７２及び１７４は、ユーザーが削除してしまったファイルである。

【０１１５】クリーンアップ動作中、アクティブファイルの中で最も古いものをブロック１１４及び１１６に再び書込む。図８に示すファイル構造１１０の場合、ブロック１３８，１４０，１４２及び１４４は最も古いアクティブ連続可変ファイル及び削除連続可変ファイルを含む。ブロック１３８，１４０，１４２及び１４４の中に見られるアクティブファイルは、ブロック１１４及び１１６に再び書込むべきファイルである。クリーンアップ動作中には、４つのブロック１３８，１４０，１４２及び１４４を２つの予約ブロック１１４及び１１６にはめ込むことができるように、圧縮動作が起こる。圧縮動作中、先にユーザーが削除したファイルはブロック１１４及び１１６に書込まれない。図８では、ブロック１３８，１４０，１４２及び１４４の中に見られる削除ファイルはファイル１６８，１７０，１７２及び１７４である。従って、それらの削除ファイル１６８，１７０，１７２及び１７４はブロック１１４及び１１６には再び書込まれない。ブロック１３８，１４０，１４２及び１４４は次に圧縮すべき領域にあるという。圧縮可能なスペースの量は、削除ファイル１６８，１７０，１７２及び１７４が占めるスペースの量である。

【０１１６】ブロック１１４，１１６，１１８及び１２０は、先にクリーンアップ動作の結果として発生したブロックである。従って、ブロック１１４，１１６，１１８及び１２０は、最前のクリーンアップ動作の結果として発生した圧縮スペースを構成しているという。

【０１１７】図８に示すファイル構造のような可変ファイル構造についてのクリーンアッププロセスは循環方式である。予約ブロックは、フラッシュメモリアレイ１１２の物理的最上部からその物理的最下部へループしてゆく場合を除いて、連続する順序である。先に述べた通り、使用可能な空きブロックの数が予約ブロックの許容数に到達すると、クリーンアップ動作が始まる。アクティブファイル及び削除ファイルを含む最も古いブロックを、まず最初に圧縮し、クリーンアップする。連続するアクティブ可変ファイルと削除可能ファイルを含む最も古いブロックは、フラッシュメモリアレイ１１２の中の予約ブロックのすぐ後に見出される。クリーンアップ動作の一部として再び書込むべきブロック及び圧縮すべきブロックの数は、それらのブロックの中の削除ファイルの数によって決まる。言いかえれば、最も古いブロック

の中の圧縮可能スペースが、１回のクリーンアップ動作の一部として再び書込まれるそれらのブロックの中のブロック数を決定する。たとえば、ブロック１３８，１４０，１４２及び１４４は４つのブロック１３８，１４０，１４２及び１４４を２つのブロック１１４及び１１６に圧縮させるのに十分な数の削除ファイルを含む。

【０１１８】ブロック１３８，１４０，１４２及び１４４をブロック１１４及び１１６に圧縮した後、ブロック１３８，１４０，１４２及び１４４を消去する。ブロック１３８，１４０，１４２及び１４４の消去後、ブロック１３８，１４０及び１４２は予約ブロックになり、ブロック１４４は空きブロックになる。

【０１１９】図８に示すファイル構造１１０は、フラッシュメモリアレイ１１２の内部のファイルの属性と場所を記憶するディレクトリを含む。ファイル構造１１０の場合、全てのディレクトリ情報はフラッシュメモリアレイ１１２のブロック１３４の中に記憶される。そこで、ファイル構造１１０のディレクトリをフラッシュメモリベースディレクトリという。

【０１２０】ファイル構造１１０のディレクトリは動的ディレクトリ再配置を要求する。動的ディレクトリ再配置とは、ファイルシステムが書込み動作と消去動作によってディレクトリをフラッシュメモリアレイ１１２の中に様々な場所へ移動させることを意味する。図８に示す例の場合、ディレクトリはフラッシュメモリアレイ１１２のブロック１３４を占める。ところが、様々な時点で、ディレクタは、ブロック１１４，１１６，１１８，１２０，１２２などを含めたフラッシュメモリアレイ１１２のその他のブロックの中のいずれかを占めることができるであろう。

【０１２１】図８に示す特定の時点においては、ディレクトリはブロック１３４にある。ブロック１３４はディレクトリフラグ１５２と、ブート記録１５４と、ブロック状態テーブル１５６（ＢＳＴ１５６ともいう）と、連係リストディレクトリエントリ１５８と、空きディレクトリスペース１６０とを含む。このブロックをディレクトリブロックともいう。

【０１２２】ディレクトリフラグ１５２は、どのブロックがファイル構造１１０に関するルートディレクトリを記憶するかを指示するデータのパターンを記憶する。本発明の一実施例においては、ディレクトリフラグ１５２は、ディレクトリフラグ１５２を含んでいるブロックがルートディレクトリを含むブロックでもあることを指示する独自のランダムでないデータパターンを記憶する。たとえば、一実施例では、ディレクトリフラグ１５２は一連の反復ＡＡｈ／５５ｈデータを記憶する。別の実施例では、ディレクトリフラグ１５２は他の何らかのランダムでないデータのシーケンスを記憶する。

【０１２３】本発明の一実施例においては、ある特定のブロックが実際にディレクトリブロックであるのか否かを判定するために補足検査を実行する。これは、ディレクトリブロック以外のブロックがディレクトリフラグ１５２のデータパターンと一致する開始データパターンを有していた場合に起こると思われる、ディレクトリブロックの場所を規定する際に誤りを回避するのを助けるために実行される。補足検査は、ディレクトリブロック１３４のブートレコード１５４の中にある検査合計データフィールドを検査することにより実行される。一実施例では、ブートレコード１５４の検査合計データフィールドは、ブートレコード１５４が有効であり且つ現在記録である場合には変化しないことから選択された特定のデータのパターンである。別の実施例においては、ブートレコード１５４の検査合計データフィールドは専用サイクル冗長性検査（「ＣＲＣ」）又は誤り修正コード（「ＥＣＣ」）データフィールドである。まず、コンピュータシステムはディレクトリフラグを探索し、次に、検査合計データフィールドを検査することになるであろう。ブートレコード１５４の検査合計データフィールドの検査から正しい結果が得られると、コンピュータシステムはフラッシュメモリアレイ１１２に関する現在有効ディレクトリブロックを実際に見出したことを知るであろう。

【０１２４】本発明の一実施例においては、ブートレコード１５４が有効でなくなるか又は現在レコードでなくなるときには、ブートレコード１５４の検査合計データフィールドの論理値１を論理値０に重ね書きするであろう。

【０１２５】ブートレコード１５４はブロック１３４の内部の、ディレクトリフラグ１５２の隣に位置している。ブートレコード１５４は特定のファイルシステムの型及び特定のファイルシステムの改訂に関する情報をレコードする。ブートレコード１５６はいくつかのファイルシステムパラメータをさらに含んでいる。それらのパラメータには、定様式化可能容量情報、予約ブロック情報、付属ディレクトリ情報、旧ディレクトリに関連する情報及びルートディレクトリに関連する情報がある。

【０１２６】定様式化可能容量情報はフラッシュメモリアレイ１１２のコード及びデータを記憶するために定様式化することができる領域の大きさである。予約ブロック情報は、どのブロックが予約ブロックであるかを識別する。先に述べた通り、予約ブロックはクリーンアップ動作及び再割付け動作の間に使用されるブロックである。付属ディレクトリ情報は、追加ディレクトリ情報の記憶のために使用されている追加ブロックの場所である。１つ又は複数の追加ブロックに記憶されている付属ディレクトリは、フラッシュメモリアレイ１１２の内部に記憶されているファイルの数がある量を越えた場合に必要になるであろう。旧ディレクトリは現在ディレクトリのバックアップコピーである。旧ディレクトリに関連するパラメータ情報は、フラッシュメモリアレイ１１２

の中の旧ディレクトリが記憶されている場所である。たとえば、図８では、旧ディレクトリはブロック１１４に記憶されている。ルートディレクトリは第１のディレクトリエントリである。ルートディレクトリに関連するパラメータ情報はフラッシュメモリアレイ１１２の中のルートディレクトリの場所である。

【０１２７】ブロック状態テーブル１５６は、どのブロックがアクティブブロックであるか、どのブロックが予約ブロックであるか及びどのブロックが欠陥ブロック又は故障ブロックであるかに関する情報を含む。一実施例では、ブロック状態テーブル１５６は各ブロックに関するサイクルカウント情報をさらに含む。サイクルカウント情報は、ある１つのブロックが消去され、再書込みされた回数を指示する。ブロック状態テーブル１５６はブロック１３４の中のブートレコード１５４の隣に位置している。

【０１２８】連係リストディレクトリエントリ１５８を主ディレクトリ１５８ともいう。主ディレクトリ１５８はブロック１３４の中のブロック状態テーブル１５６の隣に位置している。

【０１２９】主ディレクトリ１５８は一連の個別ディレクトリエントリを含む。各ディレクトリエントリはディレクトリエントリの型、すなわち、ディレクトリエントリが（１）　ファイルに関わるものであるか又は（２）　サブディレクトリに対するポインタであるかに関する情報を含む。

【０１３０】各ディレクトリファイルエントリは、ファイル名／拡張と、日時と、ファイルの型と、ファイルの長さと、フラッシュメモリアレイ１１２における物理ファイルアドレスと、シブリングポインタと、チャイルドポインタとに関する情報を含む。

【０１３１】本発明の一実施例においては、ファイル名／拡張は、ユーザーにファイルを識別させる１０バイトである。その実施例の場合、ファイル名／拡張は語，数及び文字の形をとることができる。

【０１３２】ファイルエントリ中の日時情報は、ファイルが作成された日付、もしくはファイルが最後に改訂又はセーブされた日付を指定する。

【０１３３】ファイル型情報は、ファイルがコードから成るか、ページ形コードから成るか、データから成るか又はデータ付属セグメントから成るかを表わす。

【０１３４】ファイル長さ情報はファイル又はセグメントの物理的大きさを表わす。物理的ファイルアドレス情報はフラッシュメモリアレイ１１２の中におけるファイル又はセグメントの場所を表わす。

【０１３５】シブリングポインタとチャイルドポインタは連係リストディレクトリ階層から構成される。言いかえれば、シブリングポインタとチャイルドポインタはディレクトリ構造ツリーから構成されるのである。

【０１３６】各ポインタは別のディレクトリエントリの場所を指示する。本発明の一実施例においては、ポインタはフラッシュメモリアレイ１１２の中の指示されるディレクトリエントリの絶対物理アドレスである。ところが、本発明の別の実施例では、各ポインタは、そのディレクトリを記憶しているブロックにおける増分アドレスを表わすオフセットアドレスから構成されている。

【０１３７】絶対物理アドレスから成るポインタの１例が１６進数の１２６４ＦＦＦである。これに対し、オフセットアドレスから成るポインタの１例は１６進数のＦＯＦである。ポインタのオフセットアドレスをインデックスアドレスともいう。

【０１３８】ポインタについてインデックス方式を採用する実施例では、ポインタは最小のサイズに抑えられる。また、インデックス方式によれば、クリーンアップ動作後にディレクトリを再配置してから、各ポインタアドレスを計算し直す必要はない。

【０１３９】フラッシュメモリアレイ１１２のブロック１つではディレクトリ全体を記憶しきれない場合には、ルートディレクトリにより単一の二次付属ディレクトリブロック又は２つ以上の二次付属ディレクトリブロックを指示することができる。１つ又は複数の付属ディレクトリブロックを使用するときには、各「インデックス型」ポインタは、ディレクトリエントリポインタがディレクトリブロックの中のどれを参照しているかを確定するためのインデックスを含んでいるものと考えられる。

【０１４０】本発明の一実施例では、全てのディレクトリエントリを共通の長さにしなければならないようになっており、それにより、ポインタのインデックスアドレスのサイズはさらに縮小される。この共通の長さはインデックスが参照するオフセットアドレス増分に対応する。

【０１４１】ディレクトリエントリ１５８は連係リストを形成する。ファイルシステムディレクトリツリーの全体は、ルートディレクトリエントリを開始点として、シブリングポインタ及びチャイルドポインタを使用しながらディレクトリエントリを連係リストと連鎖させてゆくことにより構成される。連係リストディレクトリツリーの例を図９及び図１０に示す。図９及び図１０に示されているディレクトリエントリは、ルートディレクトリエントリ１５７と、エントリＡ１８１と、エントリＢ１８３と、エントリＣ１８５と、エントリＤ１８９と、エントリＥ１９１と、エントリＦ１９３と、エントリＧ１９７と、エントリＨ１９９と、エントリＩ１８７と、エントリＪ１９５である。チャイルドポインタはポインタ１５９，１６１，１６５，１６９及び１７１である。また、シブリングポインタはポインタ１６３，１６７，１７１，１７３，１７５及び１７９である。

【０１４２】図９は、連係リストディレクトリエントリの論理的配列を示す。図１０は、フラッシュメモリアレイ１１２のブロック１３４の中における連係リストディ

レクトリエントリの物理的場所を示す。

【０１４３】ディレクトリエントリのどの連係リストチェーンにおいても、その最後のディレクトリエントリのシブリングポインタフィールド及びチャイルドポインタフィールドは、全てが論理値１であるＦ（１６進数）ナルデータパターンを含む。これは、そのディレクトリエントリがチェーンの終端にあることを示している。（消去されたディレクトリエントリはその全体、すなわち、その全てのフィールドにＦ（１６進数）ナルデータパターンを含む。）

【０１４４】ファイルを作成、移動又は変更するたびに、連係リストチェーンにディレクトリエントリを追加する。特定の連係リストチェーンにおける最終ディレクトリエントリのシブリングポインタフィールド又はチャイルドポインタフィールドのＦ（１６進数）ナルデータパターンは、新たに作成されるディレクトリエントリに対するシブリングポインタ又はチャイルドポインタである新たなデータパターンに重ね書きされる。フラッシュメモリアレイの個々のビットセルを論理値１から論理値０に重ね書きすることは可能であるので、この重ね書きを実行することができる。従って、ファイルが作成、移動又は変更されるにつれて、ディレクトリエントリ１５８の連係リストチェーンは成長し、空きスペース１６０の中へ移動する。言いかえれば、空きディレクトリスペース１６０は時間の経過に伴う連係リストディレクトリエントリ１５８の拡張のための余地を与える。

【０１４５】各ディレクトリエントリは、そのファイル又はサブディレクトリエントリの有効性を指示するための状態ビットをさらに有する。それらの状態ビットは、ディレクトリエントリが単にディレクトリの連係を継続してゆくだけであるが、アクティブファイル又はサブディレクトリを表わさなくなることを指示するために、０の状態に重ね書きされる。単なる連係を指示するために状態ビットが０の状態に重ね書きされた時点で、ディレクトリエントリはフラッシュメモリアレイ１１２内部の未使用スペースであると考えられる。さらに、ディレクトリエントリがファイル型エントリであり且つエントリが単にディレクトリの連係を継続してゆくだけであること指示するために状態ビットが０の状態に重ね書きされた場合には、このファイル型ディレクトリエントリと関連するフラッシュメモリアレイ１１２の中のスペースは未使用スペースであると考えられる。

【０１４６】その後の書込みのためにこの未使用スペースを解放するため、ファイルシステムのクリーンアップが必要である。フラッシュメモリブロックは全て一度に消去されるので、それぞれの未使用ディレクトリエントリを個別に再書込みすることは不可能である。動的ディレクトリ再配置はこの問題を克服するのを助ける。

【０１４７】クリーンアップ中、未使用のディレクトリエントリを圧縮する。圧縮される未使用のディレクトリエントリを伴う新たなディレクトリ連係リストは、全てルートディレクトリエントリから有効に始まらなければならない。動的ディレクトリ再配置は、別個のブロックにおける新たな連係リストを再構成することを伴う。先のディレクトリブロックは、新たなディレクトリツリーが構成されるまでバックアップとして働く。新たなディレクトリツリーは、新たな付属ディレクトリブロックと共に新たなディレクトリから構成されても良いであろう。本発明の一実施例においては、ディレクトリ再配置の一部であるディレクトリ構成の間に、新たなディレクトリをフラッシュメモリアレイ１１２の予約ブロックに直接に書込む。たとえば、フラッシュメモリアレイ１１２の予約ブロック１１６に新たなディレクトリを書込むことができるであろう。

【０１４８】クリーンアップに際しては、まず、最終的に消去されるブロックから現在活動中である全てのファイルを取出すが、それが再配置である。新たなディレクトリは旧ファイルの圧縮（すなわち、再移動）と、フラッシュメモリアレイ１１２における依然として活動中のファイルの物理的再配置の双方を反映している。本発明の一実施例では、それらの転送ファイルは旧フラッシュメモリ記憶場所から新たなフラッシュメモリ記憶場所に直接に書込まれる。

【０１４９】ブロック状態テーブル１５６は、どのブロックが空き、使用中、予約中又は欠陥ブロックであるかを追跡するために使用される。ブロック状態テーブル１５６は通常のファイル読取り動作中、通常のファイル書込み動作中及びクリーンアップ動作中にブロックを追跡する。

【０１５０】本発明の別の実施例においては、ＲＡＭ緩衝を利用する。ＲＡＭ緩衝を利用した場合、クリーンアップ動作中、ファイルがフラッシュメモリアレイ１１２の中の新たな記憶場所に書込まれる前に、即時ステップとして、パーソナルコンピュータシステムのＲＡＭにファイルを書込む。ＲＡＭ緩衝の場合には、転送ファイルはフラッシュメモリアレイ１１２の旧記憶場所から新たな記憶場所へ直接には書込まれない。その代わりに、ファイルはパーソナルコンピュータシステムのＲＡＭを通過する。

【０１５１】本発明の別の実施例では、いくつかのディレクトリエントリが付属ポインタをさらに含んでいる。付属ポインタは個々のデータファイルを２つのデータファイルに分割するために使用される。その一方のデータファイルがある１つのブロックにあるとき、他方のデータファイルは別のブロックにある。これによりデータファイルをブロック境界のあたりで分割することができる。また、個々のデータファイルはフラッシュメモリアレイ１１２の物理アドレスの終わりから始めへループすることができる。さらに、個々のデータファイルは欠陥のある、すなわち、不良のブロックのあたりで飛越すこ

とができる。

【０１５２】本発明のさらに別の実施例においては、ファイルの各ディレクトリエントリはファイル名、接尾部及びアドレスに関する二次フィールドをさらに含む。これにより、元のディレクトリエントリを放棄する必要なく、ファイルの再名前付け、編集及び追加が一度に可能になる。たとえば、データファイルを編集するときにバックアップコピーを保持する場合に接尾部再名前付けを使用できるであろう。

【０１５３】本発明の別の実施例では、図８に示すファイル構造１１０はヘッダをさらに含む。ヘッダは、たとえば、ブロック１２２にあり、その場合、ブロック１２２は空きブロックではなくなる。

【０１５４】ヘッダは、ディレクトリ及びフラッシュメモリアレイ１１２に関する情報を含むファイルである。一実施例では、ヘッダはフラッシュメモリアレイ１１２全体の未定様式化サイズと、フラッシュメモリアレイ１１２の定様式化済サイズと、フラッシュメモリアレイ１１２中のブロックの総数とに関する情報並びにフラッシュメモリアレイ１１２に関する詳細な装置情報とを含む。本発明の一実施例においては、ヘッダに記憶される詳細な装置情報は消去電圧及び書込み電圧と；消去指令及び書込み指令と；消去アルゴリズム及び書込みアルゴリズムと；個別最大循環仕様及び全チップ最大循環仕様と；フラッシュメモリアレイ１１２の読取り性能特性、書込み性能特性及び消去性能特性とを含む。

【０１５５】別の実施例では、ヘッダは１つ又は複数の代替ヘッダの場所に関する情報をさらに含む。

【０１５６】ヘッダがフラッシュメモリアレイ１１２のブロックの中の１つ、たとえばブロック１２２に記憶されているこの実施例の場合、ヘッダを周期的に再生しなければならない。ヘッダの周期的再生はヘッダを周期的に完全に消去し且つ再書込みすることを伴う。このようなヘッダの周期的再生が必要であるのは、通常、ヘッダがフラッシュメモリアレイ１１２のその他のブロックに対してコールドブロックであるためである。言いかえれば、ヘッダはフラッシュメモリアレイ１１２のその他のブロックと比べて循環頻度の少ないブロックにある。従って、ヘッダを周期的に再生しなければならないのである。

【０１５７】ヘッダをどの時点で再生しなければならないかを判定するために、ヘッダに隣接するブロックと、ヘッダの消去／プログラムサイクルの数を追跡する。主ヘッダブロックにはサイクルカウントパラメータが記憶されている。サイクルカウントはヘッダと、ヘッダに隣接するブロックの消去／書込みサイクルの数を表わす。

【０１５８】別の実施例では、ヘッダの代替（すなわち、バックアップ）コピーをフラッシュメモリアレイ１１２の別のブロック、たとえばブロック１２４に記憶する。ブロック１２４に記憶されるこの代替ヘッダは、再生動作が起こっている間にヘッダとして一時的に使用される。

【０１５９】さらに別の実施例においては、ヘッダはフラッシュメモリアレイ１１２の中にあるのではなく、パーソナルコンピュータシステムの一部を成す別個の集積回路カードにあるメモリアレイの中に記憶されている。ヘッダを記憶しているメモリアレイは「レジスタ番号でアクセスされる」メモリアレイであると考えられる。このように、ヘッダは別個の陰のアレイに記憶されるのである。

【０１６０】この陰のアレイを使用する実施例の場合、ヘッダはフラッシュメモリアレイ１１２の中にはないので、ヘッダを再生する必要はないであろう。従って、ヘッダはコールドブロックではなく、また、ホット／コールド循環に関連するサイクルカウントパラメータを含んでいる必要はないと考えられる。

【０１６１】図８に示すファイル構造１１０を参照すると、ヘッダの使用は任意であることがわかるはずである。本発明の一実施例は次のような構造によってヘッダの使用を回避している。フラッシュメモリアレイ１１２はメーカー識別コード及び装置識別コードを記憶している。本発明の一実施例では、そのようなメーカー識別コード及び装置識別コードに基づいて、パーソナルコンピュータシステムのソフトウェアは、ヘッダ型情報を記憶しているルックアップテーブルを参照し、それらの特定のコードからヘッダの型を推論することができる。

【０１６２】図１１は、フラッシュメモリアレイ２１２のファイル構造２１０を示す。ファイル構造２１０は可変サイズファイル構造（可変ファイル構造ともいう）である。フラッシュメモリアレイ２１２のブロック２１４，２１６及び２１８は「予約となるべき」ブロック（予約ブロック又は予約済ブロックともいう）である。ブロック２２０は空きとなるべきブロックである。

【０１６３】ブロック２２２，２２４，２２６及び２２８は空きブロックである。ブロック２３０は、一部にデータ又はコードを含み、一部が空いている一部空きブロックである。

【０１６４】ブロック２３２，２３３，２３６，２３８，２４０，２４２及び２４４は、ブロック２３０の一部と共に、アクティブ連続可変サイズファイル及び削除連続可変サイズファイルを構成する。一実施例では、ブロック２３２，２３３，２３６，２３８，２４０，２４２及び２４４は、ブロック２３０の一部と共に、アクティブサブディレクトリ及び削除サブディレクトリをさらに含む。ファイル２５０，２６４，２６６，２６８，２７０，２７２及び２７４はブロック２３２，２３６，２３８，２４０，２４２及び２４４の中の削除ファイルである。

【０１６５】図１１に示すファイル構造２１０の場合、ブートレコード２５４と、ブロック状態テーブル２５６

と、連係リストディレクトリエントリ２５８と、空きディレクトリエントリ２６０は、フラッシュメモリアレイ２１２の一部ではない別個のメモリ２３４に記憶される。ただし、この別個のメモリ２３４は、フラッシュメモリアレイ２１２をも含むパーソナルコンピュータシステムの一部として含まれている。

【０１６６】本発明の一実施例では、別個のメモリ２３４は従来のバイト可変電気的消去可能プログラマブル読取り専用メモリ（「ＥＥＰＲＯＭ」）である。

【０１６７】本発明の別の実施例においては、別個のメモリ２３４はスタティックＲＡＭ（「ＳＲＡＭ」）である。そのＳＲＡＭ２３４は、パーソナルコンピュータシステムの一部であるバッテリーに結合している。パーソナルコンピュータシステムへの外部からの電力が供給されなくなったとき、ＳＲＡＭ２３４に結合しているバッテリーはＳＲＡＭ２３４に電力を供給し続ける。バッテリーによりＳＲＡＭ２３４に電力が印加されると、ＳＲＡＭ２３４は、パーソナルコンピュータシステムへの給電が停止している間でもデータやコードを保持できる。バッテリーがないと、ＳＲＡＭへの給電が止まるとＳＲＡＭは全てのコードとデータを失なってしまうので、バッテリーは必要である。

【０１６８】本発明のさらに別の実施例においては、メモリ２３４はＮＶＲＡＭから構成されている。

【０１６９】図１１に示す実施例では、メモリ２３４はＥＥＰＲＯＭである。定義上、連係リストディレクトリエントリ２５８がメモリ２３４の中に見出されれば、ＥＥＰＲＯＭ２３４にディレクトリフラグを記憶する必要はない。言いかえれば、メモリ２３４には現在ディレクトリが記憶されている。

【０１７０】本発明の一実施例では、ファイル構造２１０はフラッシュメモリアレイ２１２の中に記憶されているヘッダを含む。ヘッダは、先に図８に関して論じたヘッダに含まれている情報と同様な情報を含む。図１１のファイル構造２１０に関しては、ヘッダは、ルートディレクトリがメモリ２３４の記憶場所２５８の中の一定の記憶場所にあることを示す情報をさらに含む。言いかえれば、ヘッダはルートディレクトリをメモリ２３４内部の固定物理アドレススペースにあるものとして規定する。同じように、ディレクトリフラグは不要である。

【０１７１】先の述べた通り、フラッシュメモリアレイ２１２はブロック単位でのみ消去可能であり、個々のバイトごとに消去されるのではない。これに対し、ＥＥＰＲＯＭ２３４はバイトごとに電気的に消去可能であり且つユーザー側で再書込み可能である。従って、ＥＥＰＲＯＭ２３４はバイト可変であるという。ＥＥＰＲＯＭ２３４がバイト可変であるということは、連係リストディレクトリエントリ２５８の中のファイル名、接尾部、アドレス及びポインタを個々に消去、再書込み及び変更できることを意味している。直接個別変更を先行するシブリングポインタ及びチャイルドポインタとなるようにすることにより、削除ファイルの飛越しが可能である。

【０１７２】図１１に示すファイル構造２１０の実施例では、連係リストディレクトリエントリ２５８の中の各ディレクトリエントリは連係リストディレクトリエントリ２５８の中のその他のディレクトリエントリのそれぞれとほぼ同じ大きさであると規定されている。そこで、削除したディレクトリエントリを必要に応じてランダムに再利用することができる。

【０１７３】図１１の連係リストディレクトリエントリ２５８は、その他の点では、図８の連係リストディレクトリエントリ１５８と同様である。連係リストディレクトリエントリ２５８は、先に連係リストディレクトリエントリ１５８に関連して説明してインデックス型ポインタを支援する。

【０１７４】図１１のブートレコード２５４は図８のブートレコード１５４に類似している。図１１のブートレコード２５４は特定のファイルシステムの型と、特定のファイルシステムの改訂とに関する情報を含む。ブートレコード２５４は、フラッシュメモリアレイ２１２の定様式化可能容量と、フラッシュメモリアレイ２１２に関する予約ブロック情報と、連係リストディレクトリエントリ２５８内におけるルートディレクトリの場所に関連する情報とを含むいくつかのファイルシステムパラメータをさらに含む。

【０１７５】図１１のブロック状態テーブル２５６は図８のブロック状態テーブル１５６に内容の上では類似している。図１１のブロック状態テーブル２５６はフラッシュメモリアレイ２１２のどのブロックがアクティブブロックであるか、どのブロックが予約ブロックであるか及びどのブロックが欠陥又は故障ブロックであるかに関する情報を含む。

【０１７６】図１１のフラッシュメモリアレイ２１２に関するクリーンアップ動作は、図８のフラッシュメモリアレイ１１２に関するクリーンアップ動作に類似している。図１１のフラッシュメモリアレイ２１２のクリーンアップはブロック状態テーブル２５６と関連して進行し、先に図８のフラッシュメモリアレイ１１２に関して説明したように循環方式で実行される。図１１に示すファイル構造２１０の場合、ブロック２３８，２４０，２４２及び２４４は次にクリーンアップし且つ圧縮すべきブロックである。

【０１７７】図１１の空きディレクトリスペース２６０は、将来、連係リストディレクトリエントリを拡張させてゆくことができるスペースである。ディレクトリエントリ２５８は、（１）　メモリ２３４がバイト可変ＥＥＰＲＯＭである場合はバイト可変であり、（２）　メモリ２３４がＲＡＭである場合にはビット可変であるということを理解すべきである。

【０１７８】図１２は、フラッシュメモリアレイ３０２

のファイル構造３００を示す。ファイル構造３００は、ハードディスクのディレクトリエントリやフロッピーディスクのディレクトリエントリに類似するバイト可変ディレクトリエントリ３６６を含む。バイト可変ディレクトリエントリ３６６は別個のメモリ３６０に記憶されている。

【０１７９】ファイル構造３００は、可変ファイル構造とも呼ばれる可変サイズファイル構造である。

【０１８０】フラッシュメモリ３０２のブロック３２２，３２４及び３２６は「予約となるべき」ブロック（予約済ブロック又は予約ブロックともいう）である。ブロック３２８は空きとなるべきブロックである。

【０１８１】ブロック３３０，３０４，３０６及び３０８は空きブロックである。ブロック３１０は、一部にデータ又はコードを含んでいる一部空きブロックである。

【０１８２】ブロック３１２，３１４，３１６，３１８及び３２０は、ブロック３１０の一部と共に、（１）アクティブ連続可変サイズファイル及び削除連続可変サイズファイルと、（２）　固定長アクティブサブディレクトリファイル及び固定長削除サブディレクトリファイルとを構成する。ファイル３４０，３４２，３４４，３４６，３５０及び３５４はブロック３１２，３１４，３１６，３１８及び３２０の中の削除ファイルである。ファイル３３８，３４８及び３５２はブロック３１６，３１８及び３２０の中の削除サブディレクトリファイルである。

【０１８３】図１２に示すファイル構造３１０の場合、ブート記録３６２と、ブロック状態テーブル３６４と、ディレクトリエントリ３６６は、フラッシュメモリアレイ３０２の一部ではない別個のメモリ３６０に記憶される。ただし、その別個のメモリ３６０は、フラッシュメモリアレイ３０２をも含むパーソナルコンピュータシステムの一部として含まれている。

【０１８４】本発明の一実施例では、別個のメモリ３６０は従来のバイト可変ＥＥＰＲＯＭである。別の実施例においては、別個のメモリ３６０はＳＲＡＭであり、そのＳＲＡＭ３６０はパーソナルコンピュータシステムの一部であるバッテリーに結合している。パーソナルコンピュータへの外部からの給電が停止すると、ＳＲＡＭ３６０に結合しているバッテリーはＳＲＡＭ３６０に電力を供給し続ける。

【０１８５】本発明のさらに別の実施例では、メモリ３６０はＮＶＲＡＭ（すなわち、不揮発性ＲＡＭ）から構成されている。

【０１８６】図１２に示す実施例では、メモリ３６０はバイト可変ＥＥＰＲＯＭである。定義上、ディレクトリエントリ３６６がメモリ３６０の中に見出されるのであれば、ＥＥＰＲＯＭ３６０にディレクトリフラグを記憶する必要はない。現在ルートディレクトリはメモリ３６０に記憶される。

【０１８７】一実施例においては、ファイル構造３００はフラッシュメモリアレイ３０６に記憶されているヘッダを含む。このヘッダは図８に関連して論じたヘッダに類似している。図１２のファイル構造３００の場合のヘッダは、メモリ３６０の記憶場所３６６の中の一定の記憶場所にルートディレクトリがあることを示す情報をさらに含む。同様に、ディレクトリフラグは不要である。

【０１８８】フラッシュメモリアレイ３０２はブロック単位でのみ消去可能であり、個々のバイトごとに消去することはできない。これに対し、ＥＥＰＲＯＭは電気的に消去可能であり、バイトごとにユーザー側で再書込み可能である。従って、ＥＥＰＲＯＭ３６０はバイト可変であるという。ＥＥＰＲＯＭ３６０がバイト可変であるということは、ディレクトリエントリ３６６を個別に消去、再書込み及び変更できることを意味する。

【０１８９】ファイル構造３００のディレクトリ構造３６６は、パーソナルコンピュータシステムにおけるハードディスク又はディスケットのディレクトリ構造に類似している。ディレクトリ構造３６６をディスク類似ディレクトリ構造と呼ぶ。

【０１９０】ディスク類似ディレクトリ構造３６６によって、パーソナルコンピュータシステムについて、磁気ハードディスクに基づくファイル構造との間に偽似互換性をもつオペレーティングシステムインタフェースが得られる。ファイル構造３００の場合、ディレクトリとサブディレクトリの双方の内容を記憶するために、主フラッシュメモリアレイ３０２でデータファイルを作成する。同様に、パーソナルコンピュータシステムの従来の磁気ハードディスクドライブとフロッピーディスクドライブにおいては、ディレクトリとサブディレクトリの双方の内容をディスクに記憶することになる。さらに、ディレクトリエントリ３６６は、従来の数多くのハードディスクドライブに記憶されているＤＯＳオペレーティングシステムの中に見出されるファイル割付けテーブル（「ＦＡＴ」）に類似する構造を含む。

【０１９１】図１２に示す実施例の場合、ディレクトリエントリ３６６はメモリ３６０の記憶場所３６８，３７０，３７２，３７４及び３７６に記憶されている。以下にさらに詳細に説明するように、ディスク類似ディレクトリエントリ３６６は、図８及び図１１の連係リストディレクトリエントリ１５８及び２５８とはそれぞれ異なっている。

【０１９２】図１３は、ディレクトリエントリ３６６の構造の例を示す。図１３は、ディレクトリエントリ３６６の一部であるディレクトリエントリ３０１及び３６１を示している。各ディレクトリエントリは固定長、たとえば４０バイトである。

【０１９３】ルートディレクトリエントリ３０１は８バイトの名前３１１と、３バイトのファイル拡張３１３と、１バイトの属性フラグ３１５と、１０バイトの予約

領域３１７と、２バイトの時間領域３１９と、２バイトの日付３２１と、２バイトの開始クラスタ３２３と、１バイトのクラスタ状態フラグ３２５と、４バイトのファイルサイズ領域３２７と、２バイトのポインタ３２９とを含む。

【０１９４】ファイル名３１１とファイル拡張３１３のバイトは、ルートディレクトリ３０１に対応するファイルを識別する。

【０１９５】属性フラグ３１５は8つのビット３４１，３４３，３４５，３４７，３４９，３５１，３５３及び３５５から構成されている。ビット３４１（ビット０）は、ファイルを読取り専用ファイルとして、すなわち、ファイルの編集又は消去は不可能であるとしてマークする。ビット３４３（ビット１）は、ファイルがＤＯＳに対して隠れており、ＤＯＳのディレクトリ指令及び他のいくつかのＤＯＳ指令によっては読取りできないことを指示する。ビット３４５（ビット２）は、ファイルがディレクトリ指令及び他のいくつかのＤＯＳ指令によっては読取りできないＤＯＳファイルであることを指示するシステムビットである。

【０１９６】ビット３４７（ビット３）はボリュームラベルである。ビット３４９（ビット４）はサブディレクトリポインタである。ビット３４７又は３４９がセットされている場合には、ＤＯＳはそのエントリを通常のファイルディレクトリエントリとは異なる方法で処理しなければならない。

【０１９７】サブディレクトリビット３４９がセットされている場合には、ディレクトリエントリはサブディレクトリファイルであり、サブディレクトリファイルはフラッシュメモリアレイ３０２のブロックの中の１つに記憶されることになる。

【０１９８】ビット３５１（ビット５）は、ファイルがバックアップユーティリティによりバックアップされているか否かを指示するアーカイブビットである。ビット３５３（ビット６）とビット３５５（ビット７）は、使用されない予約ビットである。

【０１９９】予約バイト３１７は予約されており、使用されない。バイト３１９及び３２１はファイルに関する日時スタンプである。バイト３２７はファイルの大きさ、すなわち、ファイルの長さを指示する。

【０２００】ルートディレクトリ３０１のバイト３２３及び３２５はルートディレクトリ３０１のファイル割付けテーブル（「ＦＡＴ」）型ビットである。開始クラスタバイト３２３はフラッシュメモリアレイ３０２の中のファイルの開始アドレスである。バイト３２５はクラスタ状態バイトと呼ばれる。クラスタ状態バイト３２５は、バイト３２３に指示されている開始アドレスから始まるファイルの状態を指示する。状態バイト３２５は、ファイルが利用可能であるか、削除ファイルであるか又は欠陥ファイルであるかを指示する。

【０２０１】バイト３２３が参照する開始クラスタは模擬人工クラスタであることを理解すべきである。ファイルディレクトリエントリ３０１が参照するファイルは実際には可変長ファイルである。ＤＯＳが記憶される磁気ハードディスクドライブ（固定又はフロッピー）に見られるディレクトリ構造に幾分か類似しているディレクトリ構造を得るために、ルートディレクトリ３０１によってクラスタファイル構造をシミュレートする。

【０２０２】ファイルサイズバイト３２７に記憶されているファイルサイズが開始クラスタ３２３のサイズより大きければ、ファイルを完全に指定するために、複数の模擬クラスタを連鎖させなければならない。この動作は次にようにして実行される。ルートディレクトリ３０１のポインタバイト３２９は図１３の付属ディレクトリエントリ３６１を指示する。付属ディレクトリエントリ３６１はファイル割付けテーブルバイト３８３及び３８５を含む。バイト３８３はクラスタバイトである。バイト３８５はクラスタ状態バイトである。

【０２０３】クラスタバイト３８３は、開始クラスタバイト３２３に記憶されているアドレスにより指示されるデータ又はコートと共に連鎖させるべきフラッシュメモリアレイ３２０内部のデータ又はコードの開始アドレスを記憶する。付属ディレクトリエントリ３６１のバイト３８５は、バイト３８３に記憶されているアドレスで始まるファイルの状態を指示する。クラスタ状態バイト３８５は、バイト３８３に記憶されているアドレスに記憶されたファイルが利用可能であるか、削除ファイルであるか又は欠陥ファイルであるかを指示する。

【０２０４】ファイルを作成するために他のデータ又はコードを共に連鎖すべきでない場合には、ディレクトリエントリ３６１のポインタ３８９は他のどのディレクトリエントリも指示しない。

【０２０５】ディレクトリエントリ３６１は単なる付属ディレクトリエントリであるので、名前バイト３７１，拡張バイト３７３，属性バイト３７５，予約バイト３７７，時間バイト３７９，日付バイト３８１及びファイルサイズバイト３８７の内容は問題にならない。

【０２０６】ディレクトリエントリの中には付属ファイルをもたないものもあるということを理解すべきである。さらに、他のいくつかのディレクトリエントリは複数の付属ファイルを有している。付属ファイルのポインタは別のファイルを指示することができる。

【０２０７】ファイル構造３００は、均一な長さのディレクトリエントリを使用して、重ね書きを経てそれらを容易に再利用可能とすることにより、メモリ３６０のバイト可変性を利用している。また、エントリのサイズが対称であるために、エントリを従来のディスクのファイル割当てテーブルのエントリとして取扱うことができる。ＦＡＴ型のクラスタ状態バイトを使用すると、ファイル構造３００に関する状態報告は相対的に単純であ

り、ＤＯＳを記憶する従来のハードディスクドライブの状態報告に類似している。

【０２０８】メモリ３６０のペアレントディレクトリエントリは、ハード（又はフロッピー）ディスクがその別個のＦＡＴインデックスを使用してファイルを参照するように、組合せディレクトリ／ＦＡＴインデックス記憶場所を参照する。

【０２０９】メモリ３６０内部の各ディレクトリ／ＦＡＴエントリは、開始クラスタとしてファイルの開始アドレスを参照することにより、ＤＯＳを模倣している。ファイルの長さが模擬クラスタより長いときには、必ず、この第１のＦＡＴエントリに人工クラスタを連鎖させる。

【０２１０】メモリ３６０に記憶されているブートレコード３６２は、模擬クラスタサイズを記憶する。ブートレコード３６２は利用可能クラスタの数と、ディレクトリエントリの最大数とをさらに記憶する。利用可能クラスタの数はフラッシュメモリアレイ３０２の定様式化可能容量である。また、ブートレコード３６２は特定のファイルシステムの型及び特定のファイルシステムの改訂に関する情報を記憶している。ブートレコード３６２は、フラッシュメモリアレイ３０２に関する予約ブロック情報と、ディレクトリエントリ３６６におけるルートディレクトリの場所に関連する情報とを含む他のいくつかのファイルシステムパラメータをさらに記憶している。

【０２１１】パーソナルコンピュータシステムのオペレーティングシステムは、メモリ３６０内部のディレクトリ／ＦＡＴエントリの数と、フォーマット上規定されている模擬クラスタサイズとを乗算した数と同等の数のセクターを含む。メモリ３６０内部のディレクトリ／ＦＡＴエントリの数は、組合わせ領域３６８，３７０，３７２，３７４及び３７６のサイズをディレクトリエントリのサイズで除算した数に等しい。

【０２１２】図１２のブロック状態テーブル３６４は内容の上では図８のブロック状態テーブル１５６と同様である。図１２のブロック状態テーブル３６４は、フラッシュメモリアレイのどのブロックがアクティブブロックであるか、どのブロックが予約ブロックであるか及びどのブロックが欠陥ブロック又は故障ブロックであるかに関する情報を含む。

【０２１３】図１２のフラッシュメモリアレイ３０２に関するクリーンアップ動作は、図８のフラッシュメモリアレイ１１２に関するクリーンアップ動作に類似している。図１２のフラッシュメモリアレイ３０２のクリーンアップはブロック状態テーブル３６４と関連して進行し、先に図８のフラッシュメモリアレイ１１２に関連して説明した通り、循環方式で実行される。図１２に示すファイル構造３００の場合、ブロック３１６，３１８及び３２０は次にクリーンアップし且つ圧縮すべきブロックである。

【０２１４】パーソナルコンピュータシステムは、新たなエントリごとに次の利用可能なスペースがどこにあるかを知るために、ディレクトリと主フラッシュメモリアレイ３０２の利用状況を追跡しなければならない。ディスクに似たセクターが欠落しているならば、これは活動循環プロセスである。

【０２１５】ディレクトリエントリ３６６、ブートレコード３６２及びブロック状態テーブル３６４のサイズがメモリ３６０のサイズを越えて拡張することは不可能である。これに対し、サブディレクトリファイルはフラッシュメモリアレイ３０２の中に記憶される。

【０２１６】通常、フラッシュメモリの情報に重ね書きすることは不可能であるため、ディレクトリファイル３６６は一定数のエントリについてあらかじめスペースを割付けておき、次に、追加エントリのための付属を指示することが必要である。

【０２１７】図１４は、フラッシュメモリアレイ４０１のファイル構造４００を示す。ファイル構造は、複数のセクターと、動的フラッシュディレクトリとを含むファイル構造である。ファイル構造４００をセクター方式ファイル構造ともいう。

【０２１８】ファイル構造４００のようなセクター方式ファイル構造では、コード又はデータはセクター単位で記憶される。セクターはパーソナルコンピュータシステムの従来のハードディスクやディスケットにおけるセクターに類似している。従来のハードディスクの場合、トラックごとのセクター数は、通常、１７である。ハードディスクによっては、クラスタごとに４つのセクターがあるものもある。他のハードディスクや高密度３．５インチディスケットでは、１つのクラスタは単一のセクターを含む。クラスタは論理的にアドレッシング可能な最小記憶単位である。

【０２１９】ファイル構造４００の場合、１つのクラスタは単一のセクターから構成されている。各セクターは５１２バイト幅であり、これは固定長である。各クラスタを割付け単位（「ＡＵ」）ともいう。

【０２２０】本発明の別の実施例では、各クラスタは２つのセクターから構成されている。さらに別の実施例においては、各クラスタは３つ以上のセクターから構成されている。

【０２２１】ファイル構造４００のセクター方式ファイル構造は、従来のディスクファイル構造のソフトウェア又はファームウェアドライバに基づくエミュレーションを単純化する助けになる。これは、ファイル構造４００のセクター方式ファイル構造がハードディスクドライブに見られるセクター方式ファイル構造に類似しているためである。

【０２２２】また、ファイル構造４００のセクター方式ファイル構造によって、クリーンアップに際して消去ブ

ロックをランダムに選択することが可能になる。このことが、今度は、ファイルの移動が不要であるときにそのような移動を少なくするのにも役立つのである。さらに、余分な循環を少なくするのにも役立つ。

【０２２３】その一方で、セクター方式ファイル構造はコードファイルの代わりとして完全連続実行を支援することができないのではあるが、連続セクターページングメモリファイル方式に従えば、大きな連続部分又はより小さなコードファイルをその場で実行させることが可能である。

【０２２４】セクター方式ファイル構造の場合、クリーンアップ中のホット／コールド消去ブロック再配分の活動管理を支援するために、個々のブロックの循環を追跡しなければならない。このような個々のブロックの循環は、本来、以下に論じるクリーンアップ／再割付け規則によって管理される。

【０２２５】フラッシュメモリアレイ４０１のブロック４１６，４２４及び４２８は「予約となるべき」ブロック（予約ブロック又は予約済ブロックともいう）である。ブロック４０８は、次のクリーンアップ動作の後に空きブロックになる空きとなるべきブロックである。

【０２２６】フラッシュメモリアレイ４０１のブロック４０４，４１４及び４２０は、コード及びデータを記憶するために使用可能なブロックである空きブロックである。ブロック４３０は、記憶場所４６４にはデータ又はコードを含み、記憶場所４６２には空きスペースを有する部分空きブロックである。

【０２２７】ブロック４０２，４０６，４１２，４１８及び４２６の中のセクターは、ブロック４３０の部分４６４にあるセクターと共に、アクティブファイル、削除ファイル、アクティブサブディレクトリ及び削除サブディレクトリを構成する。ブロック４０２の中のセクター４０３，４０５，４０９，４１１，４１３，４１５，４１７，４２３，４２５及び４２７と、ブロック４０６の中のセクター４３１，４３３，４３９，４４３，４４５及び４５３とはアクティブファイルとアクティブサブディレクトリを構成するセクターの例である。また、ブロック４１２，４１８，４２６，４２６及び４３０もアクティブファイルとアクティブサブディレクトリを構成するセクターを含む。セクター４０７，４１９，４２１，４２９，４３５，４３７，４４１，４４７，４４９，４５１，４５７，４５９，４６１，４６３，４６５，４６７，４６９，４７１，４７３，４７５，４７７，４７９，４８１，４８３，４８５，４８７，４８９及び４９１は、削除ファイルと削除サブディレクトリを含むセクターである。

【０２２８】ファイル割付けテーブル４４６の使用によって、ファイル構造４００のクラスタは論理的に連結され、１つの完全ファイルを形成する。これは、固定でディスクドライブ又はフロッピーディスクドライブを伴う従来のパーソナルコンピュータシステムで実行されていたのと同様である。すなわち、固定ディスク又はフロッピーディスクのクラスタも論理的に連結されて、１つの完全ファイルを形成する。

【０２２９】図１４では、ファイル割付けテーブル４４６はブロック４２２の中に記憶されている。ファイル割付けテーブル４４６は、ファイル構造４００が使用する動的ディレクトリ方式の一部である。ブロック４２２はディレクトリフラグ４４０と、ブートレコード４４２と、ブロック状態テーブル４４４と、ＦＡＴ４４６と、連係リストディレクトリエントリ４４８と、空きスペース４５０とを含む。従って、ブロック４２２は現在ディレクトリを記憶している。

【０２３０】ディレクトリフラグ４４０は、ブロック４２２が連係リストディレクトリエントリ４４８を含むことを指示するデータのパターンを記憶する。

【０２３１】ブートレコード４４２はディレクトリフラグに続いている。ブートレコード４４２は特定のファイルシステムの型と、特定のファイルシステムの改訂とに関する情報を含む。ブートレコード４４２はいくつかのファイルシステムパラメータをさらに含む。それらのパラメータには定様式化可能容量情報と、予約ブロック情報と、付属ディレクトリ情報と、旧ディレクトリに関連する情報と、ルートディレクトリに関連する情報とがある。

【０２３２】ブロック状態テーブル４４４はブートレコード４４２に続いている。ブロック状態テーブル４４４は、どのブロックがアクティブブロックであるか、どのブロックが予約ブロックであるか及びどのブロックが欠陥又は故障ブロックであるかに関する情報を含む。ブロック状態テーブル４４４はブロックごとのサイクルカウント情報をさらに含む。このサイクルカウント情報は、１つのブロックを何度消去し、再書込みしたかを指示する。

【０２３３】ファイル割付けテーブル４４６はブロック４２２のブロック状態テーブル４４４に続いている。ファイル割付けテーブル４４６は、連係リストディレクトリエントリ４４８の中に見出されるルートディレクトリの前に位置している。

【０２３４】連係リストディレクトリエントリ４４８は主ディレクトリ４４８とも呼ばれる。主ディレクトリ４４８は一連の個別ディレクトリエントリを含む。各ディレクトリエントリは（1）　サブディレクトリを指示するポインタ（サブディレクトリはブロック４２２以外のブロックに記憶されている）又は（2）　ファイルのディレクトリのいずれかである。図１４のディレクトリエントリ４４８は、ファイル構造４４０の場合、物理主メモリ開始アドレスの代わりにＦＡＴポインタ４４６を使用している点を除いて、図８のディレクトリエントリ１５８に類似している。図１４の各ディレクトリファイル

エントリはファイルの名前／拡張、日付、時間、ファイル型、ファイル長さ、シブリングポインタ及びチャイルドポインタに関する情報を含む。各ディレクトリファイルエントリは、１つの特定のファイルで使用する第１のクラスタの番号であるクラスタ番号をさらに含む。クラスタ番号は２バイトの大きさである。一実施例では、ファイル割付けテーブル４４６は、クラスタごとに１つずつの２バイトエントリから成るリストである。ファイル構造４００のオペレーティングシステムは第１のクラスタの番号を使用して、ファイル割付けテーブル４４６をアクセスする。そのクラスタに関わるＦＡＴエントリは、ファイル中の次のクラスタのクラスタ番号を含む。ＦＡＴ４４６の各エントリはファイルの次のクラスタに対するポインタである。オペレーティングシステムはこれを利用してファイルの次にクラスタをアクセスし、ＦＡＴ４４６の特別のマーカに達するまで、このような動作を継続してゆく。本発明の一実施例では、その特別のマーカは１６進数のＦナルである。ＦＡＴ４４６のエントリが０である場合には、こらはクラスタが使用中でないことを指示する。

【０２３５】図１４のファイル構造４００の場合、ＦＡＴ４４６の各ＦＡＴエントリはフラッシュメモリアレイ４０１の物理クラスタに対して１対１の対応を有し、最後のファイル割付けクラスタに到達するまで連鎖するＦＡＴエントリを指示する。最終ＦＡＴエントリは、依然として消去されている。すなわち、全て１のデータであるＦ（１６進数）ナルデータにより指示される。

【０２３６】図１４のファイル構造４００のディレクトリエントリは、図１４のファイル構造４００の場合、物理主メモリ開始アドレスの代わりにＦＡＴポインタを使用している点を除いて、図８のファイル構造１１０のディレクトリエントリに類似している。図１４のファイル構造４００では、不連続のファイル情報をディスクに似たＦＡＴ連鎖によって結合する。

【０２３７】ファイル構造４００が固定ディスク又はフロッピーディスクのファイル構造と異なる点の１つは、連係リストディレクトリエントリ４４８の中でチャイルドポインタとシブリングポインタを使用していることである。

【０２３８】ＦＡＴエントリと主メモリクラスタそれ自体は長さの上で対称であり、共に、短縮インデックス型アドレスから構成されている。ＦＡＴエントリの最大数は一定であり、所定のクラスタサイズについての定様式化可能容量（すなわち、ＡＵごとのセクター数）と、予約ブロックの数とを確定する。これは定様式化に際してブートレコード４４２に記録される。

【０２３９】ブロック状態テーブル４４４は、「空き」状態、「予約」状態及び「不良」状態に加えて、各ブロックについてのサイクルカウント情報を含む。ブロック状態テーブル４４４はディレクトリブロックと予約ブロックをアクセス不可能として予約して（後のクリーンアップを可能にする）、ファイルシステムがそれらのブロックに対し重ね書きを実行しようとする試みを阻止する。故障ブロックも同様に回避され、ブロック状態テーブル４４４の中に指示される。

【０２４０】クラスタは、一般に、ある１つの空きブロックの終端を越えて連続して書込まれてゆき、別の空きのブロックに書込まれる後続クラスタに連結される。この通常連続しているクラスタ使用法は、適切に指定されたコードファイルについてページング方式現場実行動作を支援することができる。後続するファイル削除は主フラッシュメモリアレイの全体にランダムな未使用部分を発生させる。

【０２４１】クリーンアップが要求されると、パーソナルコンピュータシステムは後の再書込みに備えてどのブロックをクリアすべきかを識別する。言いかえれば、クリーンアップ時、コンピュータシステムは、クリーンアップに際して再割付けし、移動させるべきファイルを有するブロックを識別する。クリーンアップ時にクリアすべきブロックの識別は２つの規則により管理される。第１の規則は、循環の最も少ないブロックを選択するというものである。これはホットブロックとコールドブロックとの循環の不均衡を減少させる。

【０２４２】ブロックのサイクルカウントに大きな矛盾が存在しない場合には、第２の規則が管理する。言いかえれば、循環の配分がコンピュータシステムにより問題としてフラグ付けされない場合には、第２の規則が管理する。第２の規則は、最大数の削除セクターを伴うブロックがクリアのために選択されるブロックであると規定する。

【０２４３】この第２の規則の根拠は、削除ファイルを含むクラスタを消去に先立ってブロックから取出す必要はないということである。削除ファイルはセーブしなければならない情報を全く含んでいないので、削除ファイルをクリーンアップ動作の一部として消去することはできない。言いかえれば、削除ファイルの再割付けの必要はないのである。従って、第２の規則を採用すると、クリーンアップ動作の一部である消去に先立ってアクティブセクター、すなわち、アクティブクラスタから取出す量は最小限に抑えられる結果となる。

【０２４４】どのブロックが最多の削除ファイルクラスタを含むかを管理するのは、本質的にはランダムなプロセスである。従って、第２の規則は、クリーンアップ方式によって、クリアすべきブロックに関してブロックの選択の大部分がランダムになるということを意味する。この本質的にランダムなブロック選択はクリーンアップ効率を向上させるのを助けると同時に、総体的な循環の不一致を最小に抑える。

【０２４５】上記の第１と第２の規則が合わせてフラッシュメモリアレイ４０１内部の循環配分を管理する。こ

れら２つの規則は合わせて循環の差異を最小限に抑えるのを助ける。従って、ファイル構造４００のクリーンアップは本質的にランダムである。

【０２４６】図１４のファイル構造４００の場合、クリーンアップ時に新たなディレクトリ（及び新たな付属ディレクトリ）は新たなＦＡＴと共に予約ブロックの中に構成される。

【０２４７】一実施例では、ファイル構造４００はフラッシュメモリアレイ４０１に記憶されているヘッダを含む。ヘッダは図８のファイル構造１１０に関して論じたヘッダに含まれている情報に類似する情報を含む。

【０２４８】図１５は、フラッシュメモリアレイ５０１のファイル構造５００を示す。ファイル構造５００は複数のセクターを含むファイル構造であり、セクター方式ファイル構造と呼ばれる。

【０２４９】セクター方式ファイル構造５００では、データ又はコードはセクター単位で記憶される。図１５の実施例では、クラスタごとに１つのセクターがあり、各セクターは５１２バイトである。

【０２５０】ブロック５０８，５１６及び５２４はファイル構造５００の予約ブロックである。ブロック５０８は空きとなるべきブロックである。

【０２５１】ブロック５０４，５１４及び５２０は空きブロックである。ブロック５３０は記憶場所５６４にデータ又はコードを記憶し、記憶場所５６２には空きスペースを有する一部空きブロックである。

【０２５２】ブロック５０２，５０６，５１２，５１８及び５２６の中のセクターは、ブロック５３０の部分５６４の中のセクターと共に、アクティブファイル及び削除ファイルと、アクティブサブディレクトリ及び削除サブディレクトリとを構成する。ブロック５０２の中のセクター５０３，５０５，５０９，５１１，５１３，５１５，５１７，５２３，５２５及び５２７と、ブロック５０６の中のセクター５３１，５３３，５３９，５４３，５４５及び５５３はアクティブファイルセクターと、アクティブサブディレクトリセクターの例である。ブロック５１２，５１８，５２６及び５３０もアクティブファイルセクター及びアクティブサブディレクトリセクターを含む。セクター５０７，５１９，５２１，５２９，５３５，５３７，５４１，５４７，５４９，５５１，５５７，５５９，５６１，５６３，５６５，５６７，５６９，５７１，５７３，５７５，５７７，５７９，５８１，５８３，５８５，５８７，５８９及び５９１は、削除ファイルと削除サブディレクトリを含むセクターである。

【０２５３】図１５に示すファイル構造５００の場合、ブートレコード５４２と、ブロック状態テーブル５４４と、ファイル割付けテーブル５４６と、連係リストディレクトリエントリ５４８と、空きディレクトリスペース５５０とは別個のメモリ５３４に記憶されている。ただし、別個のメモリ５３４は、フラッシュメモリアレイ５０１をも含むパーソナルコンピュータシステムの一部として含まれたいる。

【０２５４】本発明の一実施例においては、メモリは従来のバイト可変ＥＥＰＲＯＭである。

【０２５５】別の実施例では、別個のメモリ５３４はバックアップとしてのバッテリーに結合するＳＲＡＭである。

【０２５６】さらに別の実施例では、別個のメモリ５３４はＮＶＲＡＭである。図１５に示す実施例においては、メモリ５３４はＥＥＰＲＯＭである。定義上、連係リストディレクトリエントリ５４８がメモリ５３４の中に見出されるとすれば、ＥＥＰＲＯＭ５３４にディレクトリフラグを記憶する必要はない。言いかえれば、メモリ５３４には現在ディレクトリが記憶されている。

【０２５７】一実施例では、ファイル構造５００はフラッシュメモリアレイ５０１の中に記憶されているヘッダを含む。ヘッダは図８に関して論じたヘッダに含まれている情報と類似する情報を記憶する。

【０２５８】図１５のファイル構造５００は、別個のメモリ５３４の固定記憶場所にディレクトリエントリ５４８を有するという点を除いて、図１４のファイル構造４００に類似している。すなわち、、ファイル構造５００はメモリ５３４に関してディレクトリフラグを有しておらず、最大数のディレクトリエントリが存在しているのである。

【０２５９】ランダムな連係リストディレクトリ修正を簡略化するために、ディレクトリエントリ５４８も同じように長さの上で対称である。ランダムな、相対的に効率の良いブロック再割付けの間には、循環管理のためにブロック状態テーブル５４４を使用する。他の全てのディレクトリ／ＦＡＴの動作は図１４のファイル構造４００の場合の動作に類似している。

【０２６０】図１６は、フラッシュメモリアレイ６０１のファイル構造６００を示す。ファイル構造６００は複数のセクターを含むファイル構造であり、セクター方式ファイル構造と呼ばれる。

【０２６１】セクター方式ファイル構造６００では、データ又はコードはセクター単位で記憶される。図１６の実施例では、クラスタごとに１つのセクターがあり、各セクターは５１２バイトである。ブロック６０８，６１６及び６２４はファイル構造６００の予約ブロックである。ブロック６０８は空きとなるべきブロックである。ブロック６０４，６１４及び６２０は空きブロックである。ブロック６３０は記憶場所６６４にデータ又はコードを含み、記憶場所６６２には空きスペースを有する一部空きブロックである。

【０２６２】ブロック６０２，６０６，６１２，６１８及び６２６の中のセクターは、ブロック６３０の部分６６４にあるセクターと共に、アクティブファイル及び削

除ファイルと、アクティブサブディレクトリ及び削除サブディレクトリとを構成する。ブロック６０２の中のセクター６０３，６０５，６０９，６１１，６１３，６１５，６１７，６２３，６２５及び６２７と、ブロック６０６の中のセクター６３１，６３３，６３９，６４３，６４５及び６５３とはアクティブファイルセクターとアクティブサブディレクトリセクターの例である。ブロック６１２，６１８，６２６及び６３０もアクティブファイルセクターと、アクティブサブディレクトリセクターを含む。セクター６０７，６１９，６２１，６２９，６３５，６３７，６４１，６４７，６４９，６５１，６５７，６５９，６６１，６６３，６６５，６６７，６６９，６７１，６７３，６７５，６７７，６７９，６８１，６８３，６８５，６８７，６８９及び６９１は、削除ファイルと削除サブディレクトリを含むセクターである。

【０２６３】図１６に示すファイル構造６００の場合、ブート記録６４２と、ブロック状態テーブル６４４と、ファイル割付けテーブル６４６と、ディレクトリエントリ６４８とは、フラッシュメモリアレイ６０１の一部ではない別個のメモリ６３４に記憶されている。ただし、この別個のメモリ６３４は、フラッシュメモリアレイ６０１をも含むパーソナルコンピュータシステムの一部として含まれている。

【０２６４】本発明の一実施例においては、別個のメモリ６３４は従来のバイト可変ＥＥＰＲＯＭである。別の実施例では、別個のメモリ６３４はバックアップとしてのバッテリーに結合するＳＲＡＭである。さらに別の実施例ではメモリ６３４はＮＶＲＡＭである。

【０２６５】図１６に示す実施例においては、メモリ６３４はＥＥＰＲＯＭである。定義上、メモリ６３４にディレクトリエントリ６４８が見出されるのであれば、ＥＥＰＲＯＭ６３４にディレクトリフラグを記憶する必要はない。言いかえれば、メモリ６３４には現在ディレクトリが記憶されているのである。

【０２６６】一実施例では、ファイル構造６００はフラッシュメモリアレイ６０１の中に記憶されているヘッダを含む。このヘッダは、図８に関して論じたヘッダに含まれている情報に類似する情報を記憶する。

【０２６７】図１６のファイル構造６００は、（１）パーソナルコンピュータシステムにおけるハードディスク又はディスケットのディレクトリ構造に類似するディレクトリ構造６４８を有することと、（２）　別個のメモリ６３４にディレクトリエントリ６４８を記憶していることを除いて、図１４のファイル構造４００に類似している。ファイル構造６００はメモリ６３４に関してディレクトリフラグを有していない。

【０２６８】図１６のファイル構造６００は、このように、ディスク類似ディレクトリ構造を有しているのである。この実現形態は、単純に、磁気ディスクのディレクトリをＦＡＴの規約に従っている。バイト可変性があるため、ディレクトリ及びＦＡＴのセクター利用／連鎖情報を直接に重ね書きすることが可能である。一次フラッシュ特有アスペクトは、消去ブロックレベルでまず消去して始めて論理セクターを重ね書きができるということになっている。従って、クリーンアップは図８のファイル構造１１０の規約に従うのであるが、サブディレクトリファイルエントリ方式と、他の大半の制約や特性は図１２のファイル構造３００に類似している。図１６のファイル構造６００は従来の磁気ディスクの動作に類似する方式で動作し、ディスククラスタに類似するファイルやサブディレクトリのクラスタの読取り及び書込みを実行する。ブロック状態テーブル６４４の利用とクリーンアップ動作は重要なフラッシュ特有の要素である。好ましい実施例においては、従来のディスクでは典型的である「バックアップディレクトリ／ＦＡＴ」方式を採用する。

【０２６９】本発明の好ましい実施例は、固有のフラッシュ循環管理によって装置の信頼性の向上を促進する。また、本発明の好ましい実施例は、固有のバックアップコピー作成、すなわち、アクティブファイル及び／又はディレクトリ情報の重ね書きの回避によって、ファイルシステムの信頼性の向上を促進する。

【０２７０】本発明の好ましい実施例は（１）　ファームウェアに制御コードを埋込み且つ（２）　クリーンアップ中の直接ブロック間ファイル／ディレクトリ転送のために予約ブロックを使用することにより、システムＲＡＭに課される要件をできる限り緩和する。

【０２７１】本発明の好ましい実施例は、連係リストディレクトリ構造の使用によって、フラッシュメモリスペースの浪費を最小限に抑える。

【０２７２】本発明の可変ファイルの実施例によれば、フラッシュアレイの小さな管理可能部分のクリーンアップを経て再割付けが可能である。

【０２７３】セクター方式／動的ディレクトリを採用する実施例は、ブロック状態テーブルを使用するのであれば、性能、信頼性及び効率の向上を促進する。ブロック状態テーブルは（１）　ホットスポットが他の装置及び装置内のブロック（それらが存在する場合）に対して消去及び書込み性能の劣化を発生させるのを阻止する一方で、装置の信頼性をできる限る高めるための均一なフラッシュ装置とブロックの循環、並びに（２）　信頼しうる急速なクリーンアップ及び「バックグラウンドタスク」消去のための予約ブロックの追跡を支援する。

【０２７４】以上の明細書においては、本発明をその特定の実施例に関連させて説明したが、特許請求の範囲に記載したような本発明のより広い範囲の趣旨から逸脱せずに様々な変形や変更を実施しうることは自明であろう。従って、明細書と図面は限定的な意味ではなく、例示を目的とするものとしてみなされるべきである。

【図面の簡単な説明】

【図１】従来のオペレーティングシステムの論理的編成を示す図。

【図２】フラッシュアレイと、コンピュータシステムのＲＡＭに記憶されているファイルシステムドライバとを含むパーソナルコンピュータシステムを示す図。

【図３】（１）　フラッシュメモリアレイと、（２）ＲＯＭ　ＢＩＯＳ及びＲＯＭＢＩＯＳ拡張部分の中に含まれているファイルシステムドライバと、（３）　ファイルを転送し且つファイルのディレクトリを更新するＲＡＭバッファとを含むパーソナルコンピュータシステムを示す図。

【図４】（１）　フラッシュメモリアレイと、（２）ＲＯＭ　ＢＩＯＳ及びＲＯＭＢＩＯＳ拡張部分に含まれているファイルシステムドライバと、（３）　別個のシステム制御装置とを含むパーソナルコンピュータシステムを示す図。

【図５】入出力マッピング方式のフラッシュメモリアレイを示す図。

【図６】ページング形マッピング方式のフラッシュメモリアレイ構造を示す図。

【図７】直接メモリマッピング方式のフラッシュメモリアレイ構造を示す図。

【図８】可変ファイル構造及び動的ディレクトリを有するフラッシュＥＥＰＲＯＭを示す図。

【図９】連係リストディレクトリの１例の論理的配列を示す図。

【図１０】連係リストディレクトリの１例の論理的場所を示す図。

【図１１】可変ファイル構造と、ディレクトリを記憶する別個のＥＥＰＲＯＭとを有するフラッシュＥＥＰＲＯＭを示す図。

【図１２】可変ファイル構造と、ディスク類似ディレクトリを記憶する別個のＥＥＰＲＯＭとを有するフラッシュＥＥＰＲＯＭを示す図。

【図１３】２つのディレクトリ／ＦＡＴエントリを示す図。

【図１４】セクター方式ファイル構造及び動的ディレクトリを有するフラッシュＥＥＰＲＯＭを示す図。

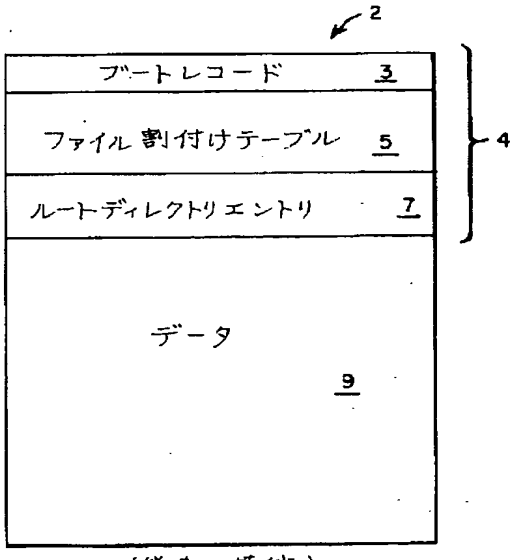
【図１５】セクター方式ファイル構造と、ディレクトリを記憶する別個のＥＥＰＲＯＭとを有するフラッシュＥＥＰＲＯＭを示す図。

【図１６】セクター方式ファイル構造と、ディスク類似ディレクトリを記憶する別個のＥＥＰＲＯＭとを有するフラッシュＥＥＰＲＯＭを示す図。

【符号の説明】

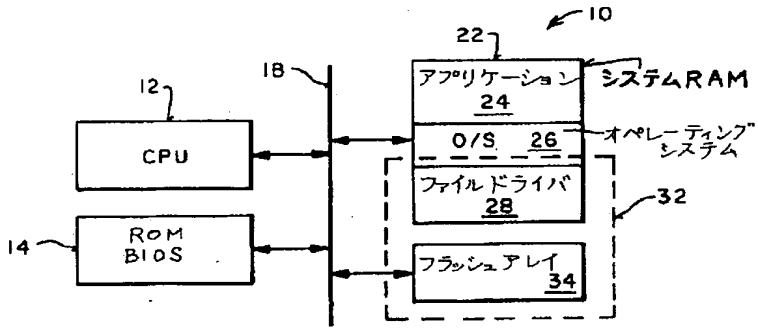
１０　パーソナルコンピュータシステム
１２　中央処理装置
３２，５８，９４　ファイル構造
３４，６４，９６　フラッシュメモリアレイ
９２　制御装置

【図１】

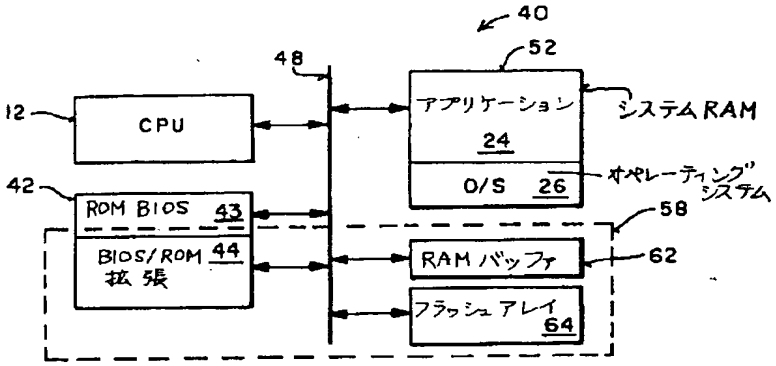


（従来の技術）

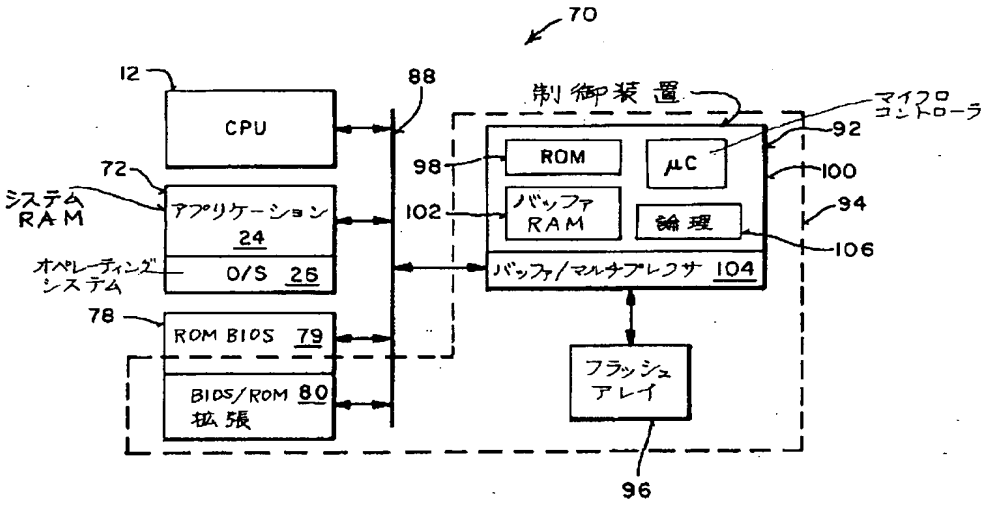
【図２】



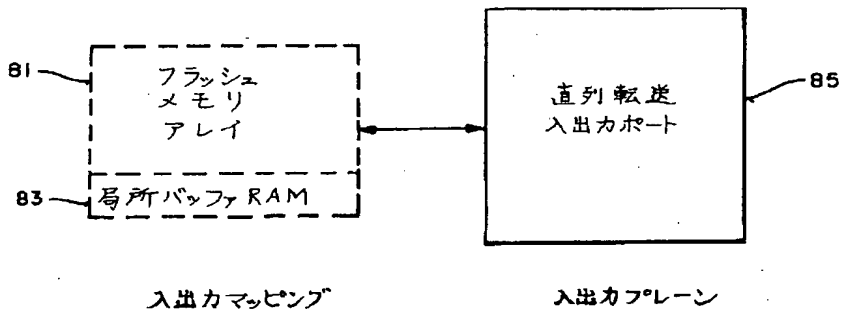
【図３】


40
48
52
12
CPU
アプリケーション
24
システムRAM
O/S 26
オペレーティング"システム
42
ROM BIOS 43
58
BIOS/ROM 44
拡張
RAMバッファ
62
フラッシュアレイ
64

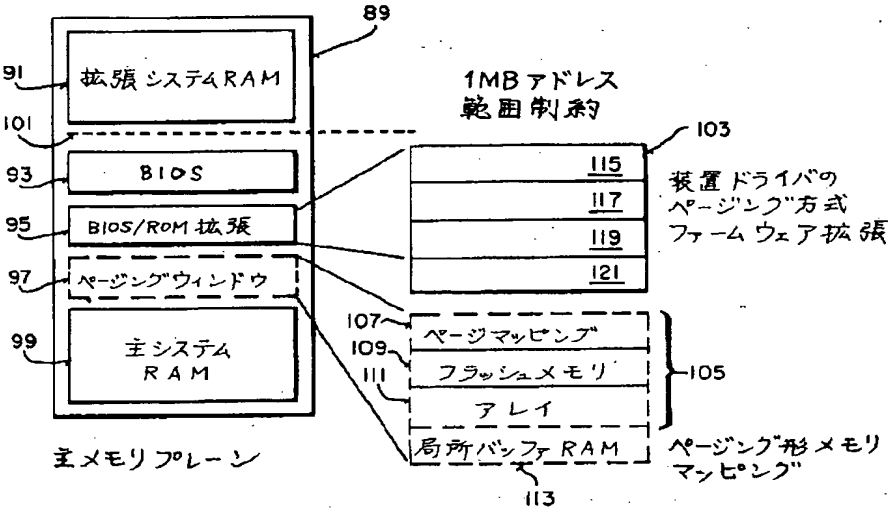
【図４】


70
12
88
制御装置
マイクロコントローラ
CPU
92
72
98
ROM
μC
100
システムRAM
アプリケーション
24
102
バッファRAM
論理
94
106
オペレーティングシステム
O/S 26
バッファ/マルチプレクサ 104
78
ROM BIOS 79
フラッシュアレイ
BIOS/ROM 80
拡張
96

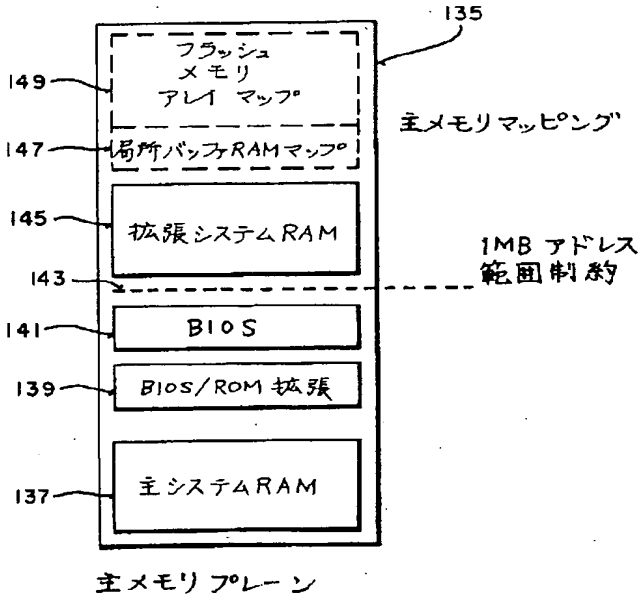
【図５】


81
フラッシュメモリアレイ
85
直列転送入出力ポート
83
局所バッファRAM
入出力マッピング
入出力プレーン

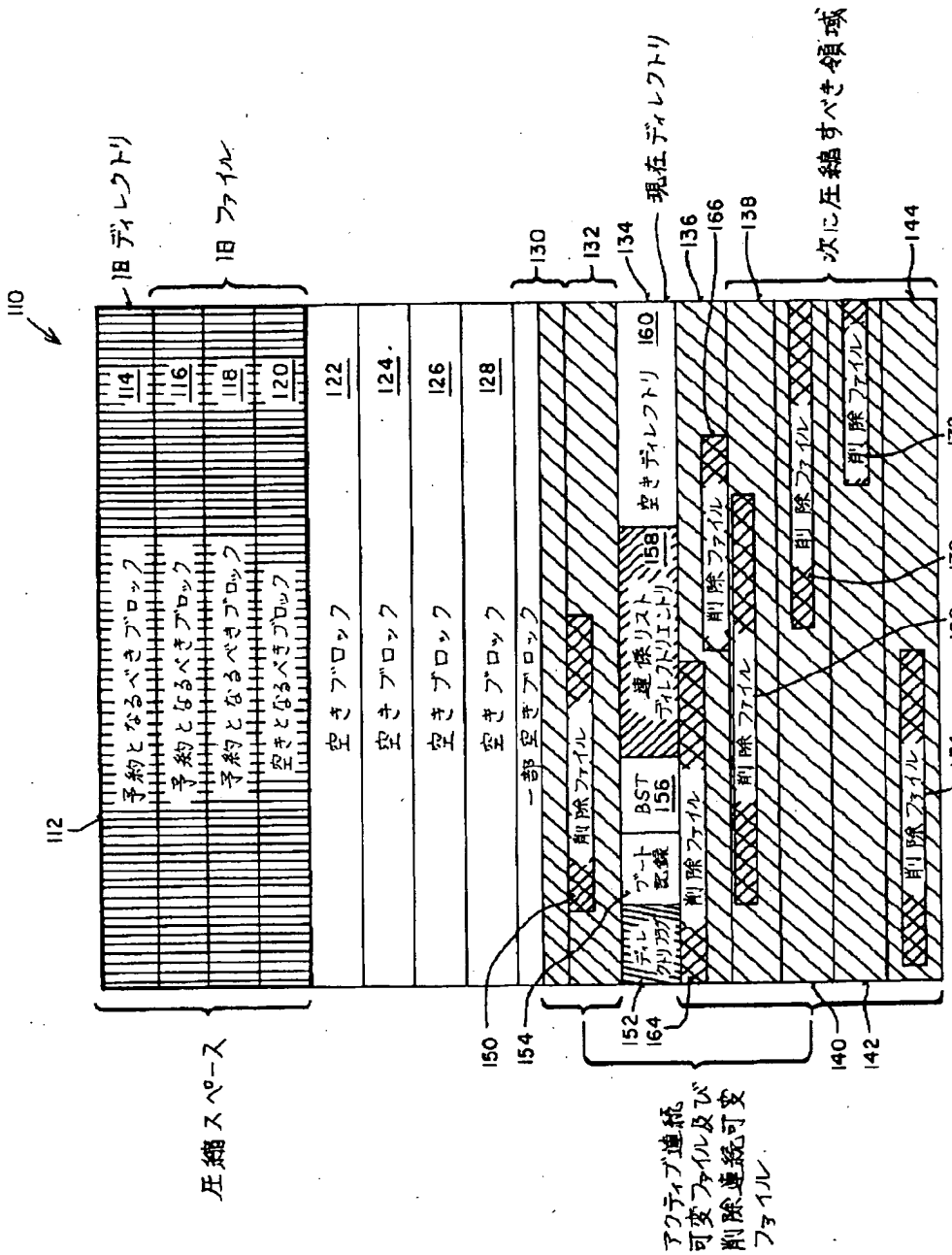
【図６】


89
91
拡張システムRAM
101
93
BIOS
95
BIOS/ROM 拡張
97
ページングウィンドウ
99
主システム RAM
主メモリプレーン
1MBアドレス 範囲制約
103
115
117
119
121
装置ドライバの ページング方式 ファームウェア拡張
107
ページマッピング
109
フラッシュメモリ
111
アレイ
105
局所バッファRAM
113
ページング形メモリ マッピング

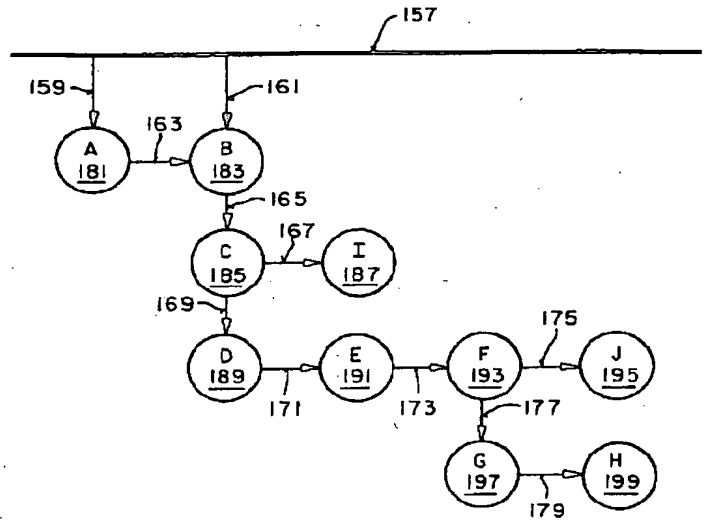
【図７】


135
149
フラッシュ メモリ アレイ マップ
147
局所バッファRAMマップ
主メモリマッピング
145
拡張システムRAM
143
1MB アドレス 範囲制約
141
BIOS
139
BIOS/ROM 拡張
137
主システムRAM
主メモリプレーン

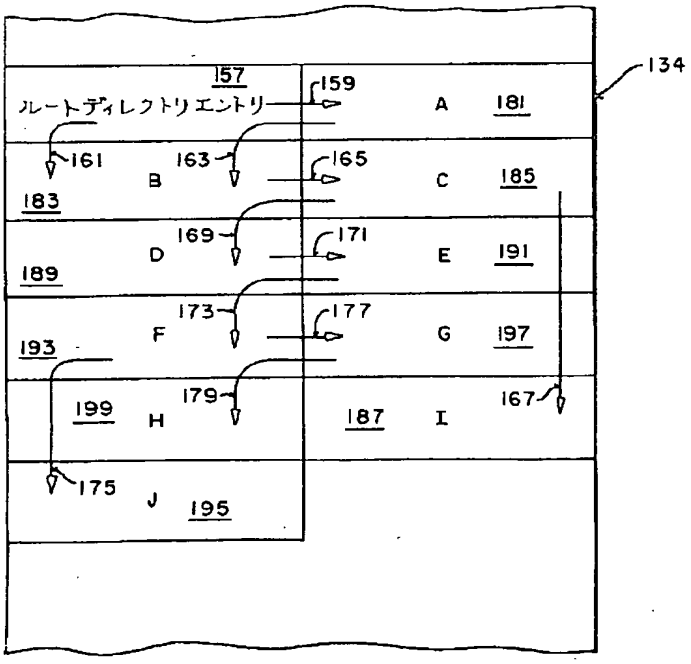
【図８】


110
18 ディレクトリ
18 ファイル
112
予約となるべきブロック 114
予約となるべきブロック 116
予約となるべきブロック 118
空きとなるべきブロック 120
空きブロック 122
空きブロック 124
空きブロック 126
空きブロック 128
130
132
134
現在ディレクトリ
136
166
138
次に圧縮すべき領域
144
一部空きブロック
削除ファイル
ブート記録
BST 156
削除ファイル
デレクトリエントリ
連係リスト 158
空きディレクトリ 160
削除ファイル
削除ファイル
削除ファイル
削除ファイル
172
170
168
174
150
154
152
164
140
142
圧縮スペース
アクティブ連続可変ファイル及び削除連続可変フォイル

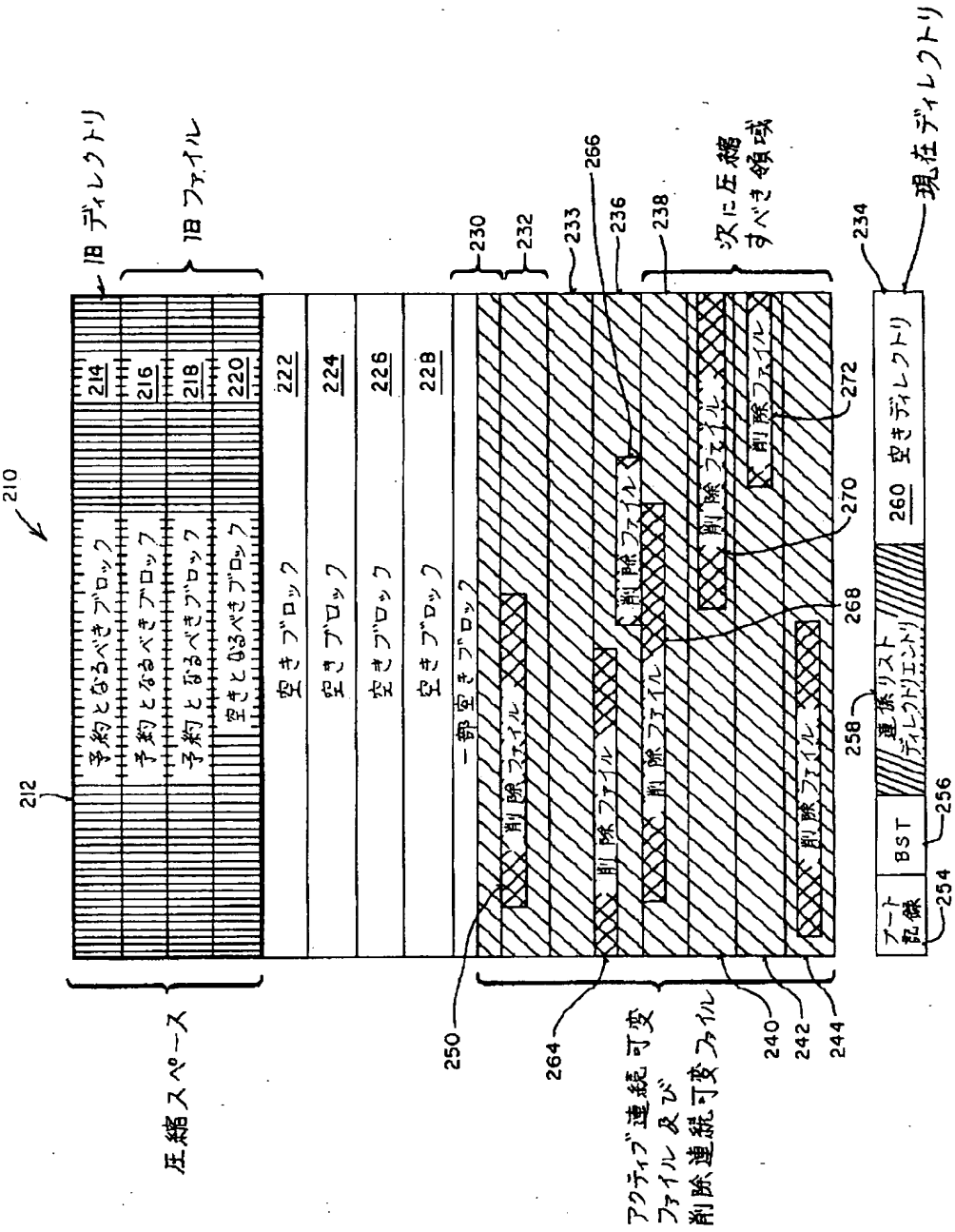
【図９】


157
159
161
163
A
181
B
183
165
C
185
167
I
187
169
D
189
E
191
F
193
175
J
195
171
173
177
G
197
H
199
179

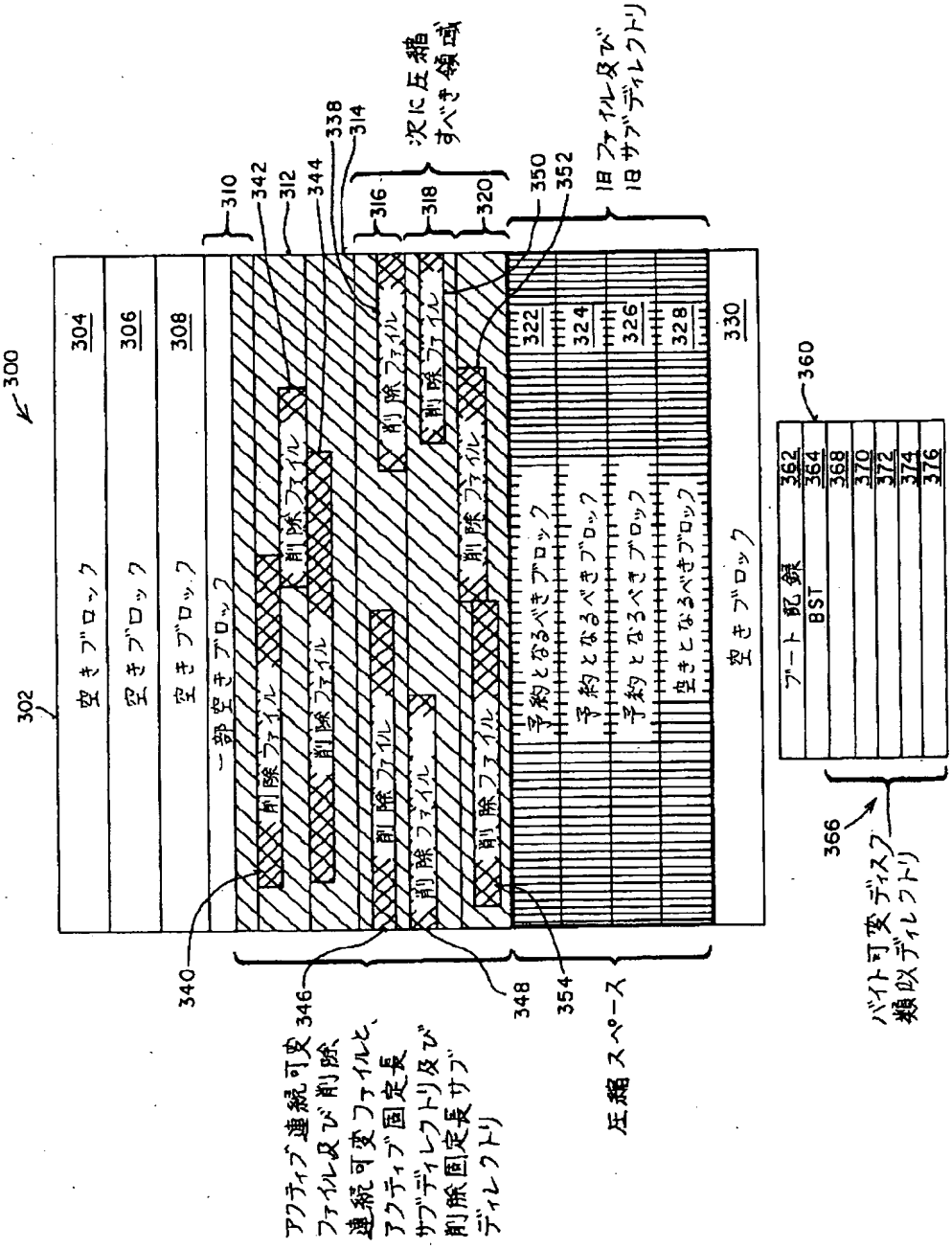
【図１０】


134
157
ルートディレクトリエントリ
159
A
181
161
163
165
B
183
C
185
169
171
D
189
E
191
173
177
F
193
G
197
199
H
179
187
I
167
175
J
195

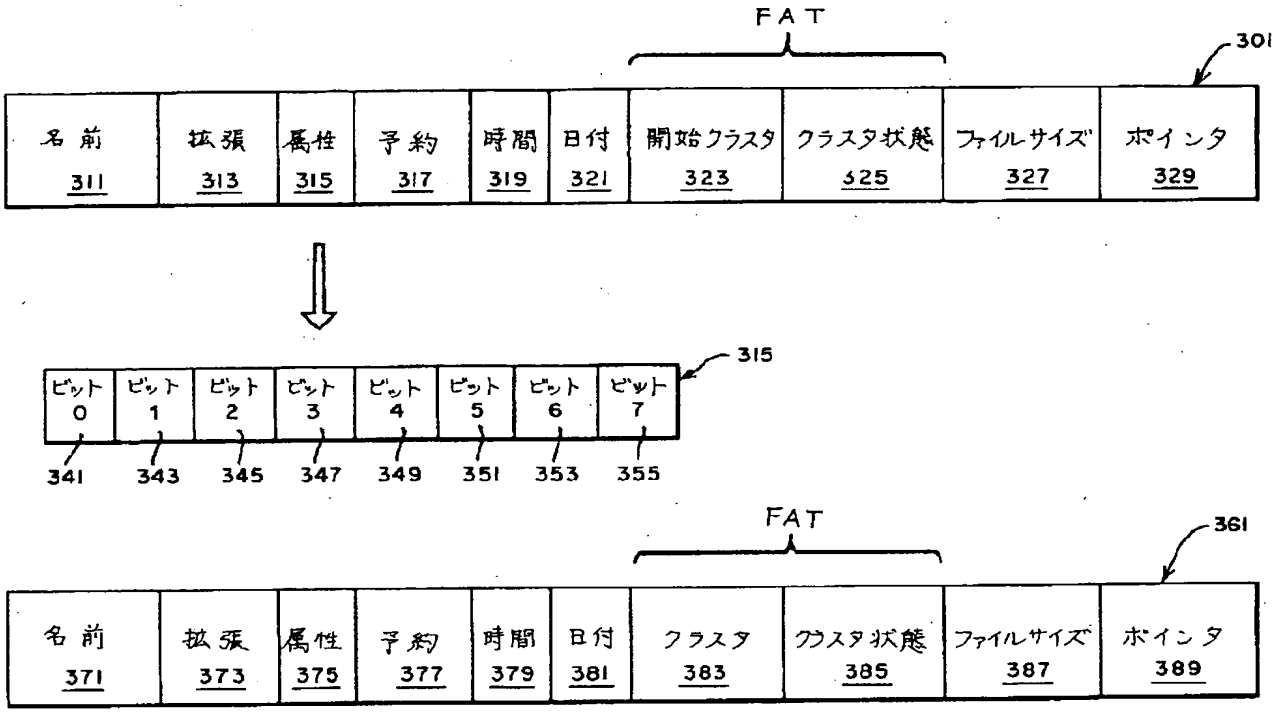
【図１１】


210
212
旧 ディレクトリ
旧 ファイル
214
216
218
220
予約となるべきブロック
予約となるべきブロック
予約となるべきブロック
空きとなるべきブロック
222
224
226
228
空きブロック
空きブロック
空きブロック
一部空きブロック
230
232
233
266
236
238
次に圧縮すべき領域
234
現在ディレクトリ
削除ファイル
削除ファイル
削除ファイル
削除ファイル
削除ファイル
272
270
268
258
260 空きディレクトリ
連続リスト
ディレクトリエントリ
256
BST
254
アート
乱候
圧縮スペース
250
264
アクティブ連続可変
ファイル及び
削除連続可変ファイル
240
242
244

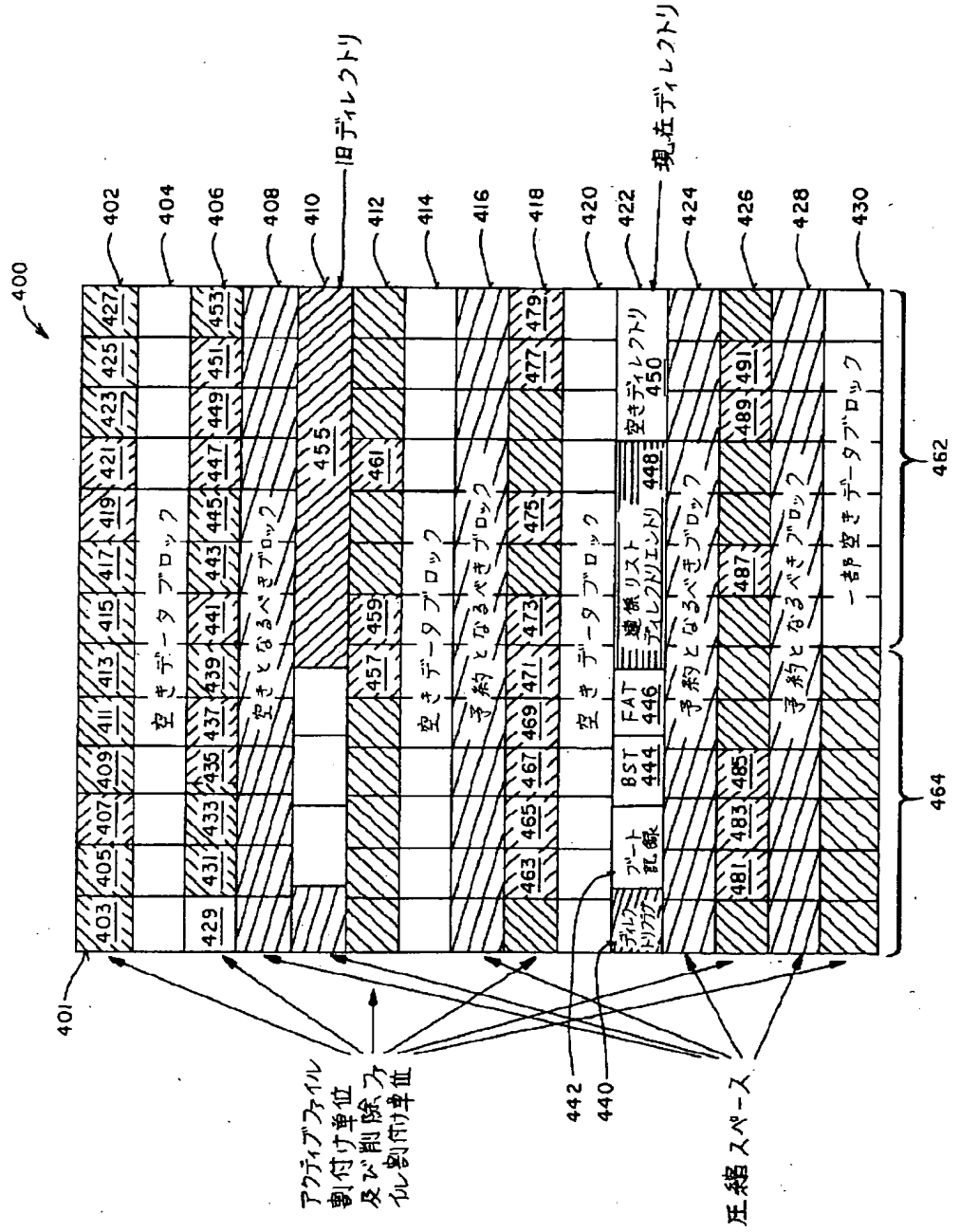
【図１２】


300
302
空きブロック 304
空きブロック 306
空きブロック 308
一部空きブロック
削除ファイル
削除ファイル
削除ファイル
削除ファイル
削除ファイル
削除ファイル
310
342
312
344
338
314
316
318
320
次に圧縮すべき領域
350
352
旧ファイル及び旧サブディレクトリ
予約となるべきブロック
予約となるべきブロック
予約となるべきブロック
空きとなるべきブロック
322
324
326
328
空きブロック 330
340
346
348
354
アクティブ連続可変ファイル及び削除、連続可変ファイルと、アクティブ固定長サブディレクトリ及び削除固定長サブディレクトリ
圧縮スペース
360
ブート記録 362
BST 364
368
370
372
374
376
366
バイト可変ディスク類似ディレクトリ

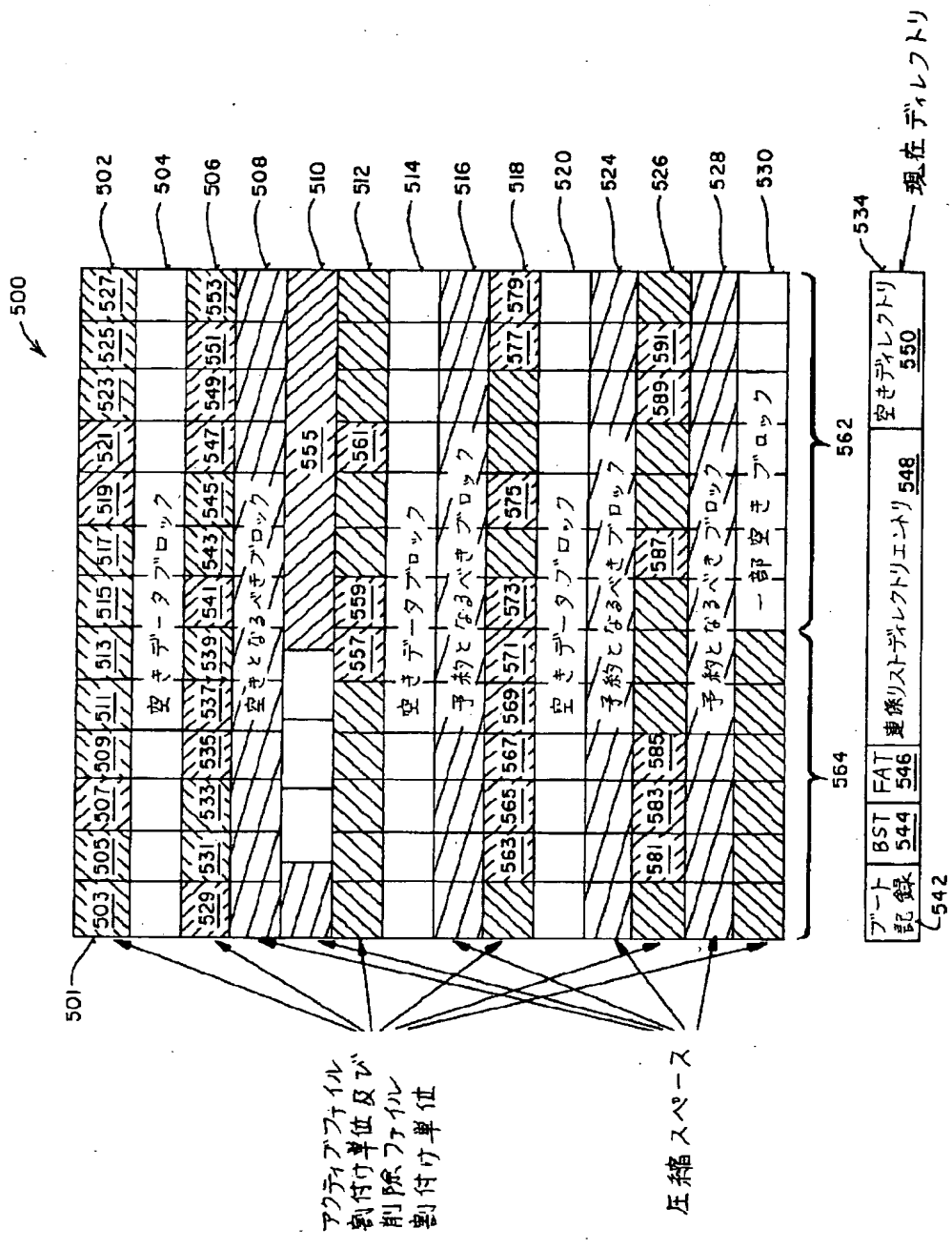
【図１３】


FAT
301
名前 311
拡張 313
属性 315
予約 317
時間 319
日付 321
開始クラスタ 323
クラスタ状態 325
ファイルサイズ 327
ポインタ 329
315
ビット 0
ビット 1
ビット 2
ビット 3
ビット 4
ビット 5
ビット 6
ビット 7
341
343
345
347
349
351
353
355
FAT
361
名前 371
拡張 373
属性 375
予約 377
時間 379
日付 381
クラスタ 383
クラスタ状態 385
ファイルサイズ 387
ポインタ 389

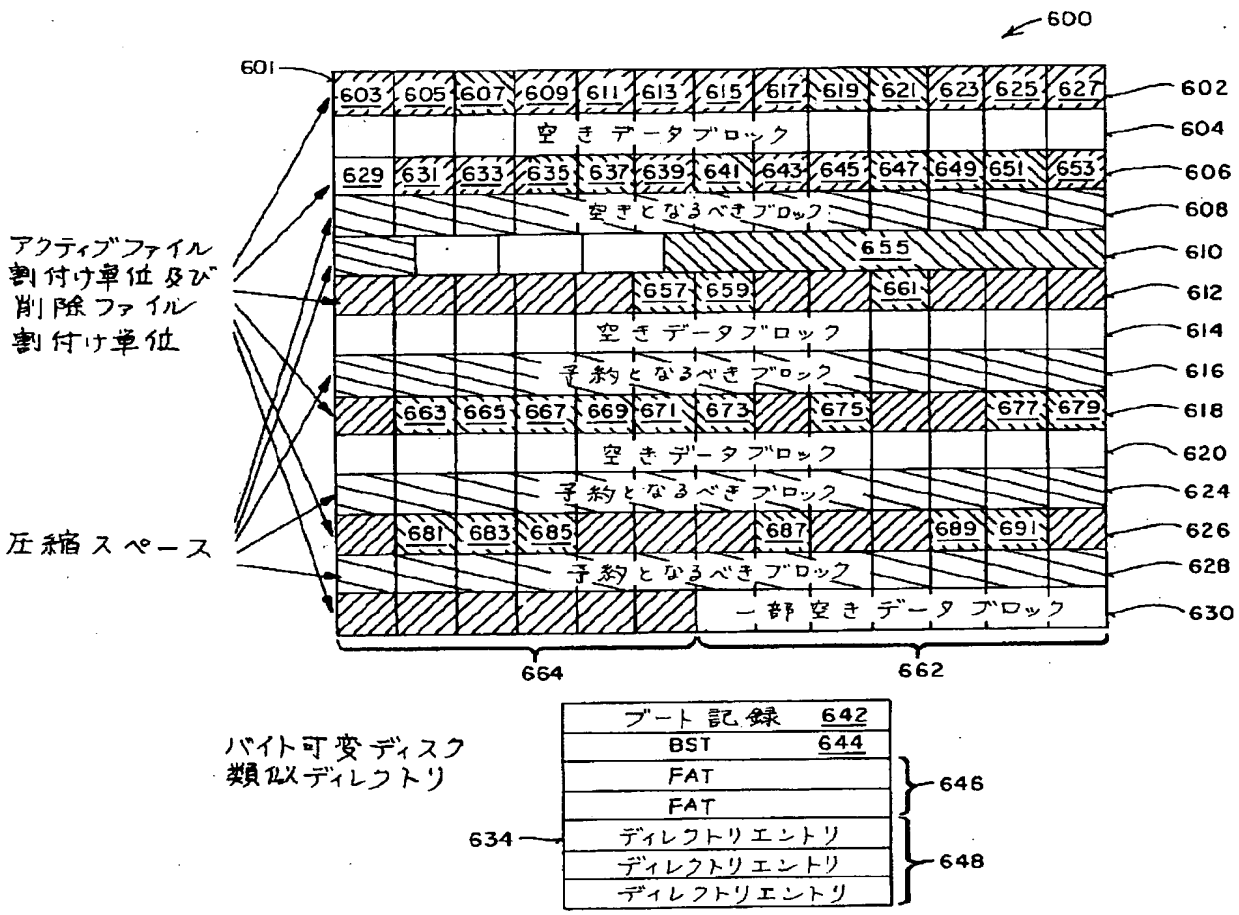
【図１４】


400
401
402
404
406
408
410
旧ディレクトリ
412
414
416
418
420
422
現在ディレクトリ
424
426
428
430
空きデータブロック
空きとなるべきブロック
455
461
457
459
予約となるべきブロック
475
473
空きデータブロック
477
479
空きディレクトリ 450
使用リスト ディレクトリエントリ 448
FAT 446
BST 444
ブート 記録
ディレクトリファイル
予約となるべきブロック
487
489
491
一部空きデータブロック
462
464
429
431
433
435
437
439
441
443
445
447
449
451
453
403
405
407
409
411
413
415
417
419
421
423
425
427
463
465
467
469
471
481
483
485
アクティブファイル割付け単位及び削除ファイル割付け単位
442
440
圧縮スペース

【図１５】


500
501
502
504
506
508
510
512
514
516
518
520
524
526
528
530
空きデータブロック
空きとなるべきブロック
555
561
空きデータブロック
予約となるべきブロック
575
空きデータブロック
予約となるべきブロック
587
591
予約となるべきブロック
一部空きブロック
562
564
534
現在ディレクトリ
空きディレクトリ 550
ブート記録 542
BST 544
FAT 546
更新リストディレクトリエントリ 548
アクティブファイル割付け単位及び削除ファイル割付け単位
圧縮スペース

【図１６】



フロントページの続き

(72)発明者　デイル・ケイ・エルバート
アメリカ合衆国　95643　カリフォルニア州・ケルセイ・キャトバード　ヒル　レイン・5900

(72)発明者　マーカス・エイ・レヴィ
アメリカ合衆国　95610　カリフォルニア州・シトラス　ハイツ・ミカ　ウェイ・8441